SYSTEM ANALYSIS

FOR THE

HUNTSVILLE OPERATIONAL SUPPORT CENTER

DISTRIBUTED COMPUTER SYSTEM

ANNUAL REPORT
MSSU-EIRS-EE-83-6
May 1982 - June 1983

Submitted by:

F. M. Ingels, Principal Investigator

Mississippi State University
Electrical Engineering Department
Mississippi State, MS 39762
(601) 325-3912

Submitted to:

NASA/MSFC, Alabama
Technical Monitor: Frank Emmens, EB 32
(205) 453-4629

NAS8-34906

July, 1983

SYSTEM ANALYSIS

FOR THE

HUNTSVILLE OPERATIONAL SUPPORT CENTER

DISTRIBUTED COMPUTER SYSTEM

ANNUAL REPORT
MSSU-EIRS-EE-83-6
May 1982 - June 1983

Submitted by:

F. M. Ingels, Principal Investigator

Mississippi State University
Electrical Engineering Department
Mississippi State, MS 39762
(601) 325-3912

Submitted to:

NASA/MSFC, Alabama
Technical Monitor: Frank Emmens, EB 32
(205) 453-4629

NAS8-34906

July, 1983

SYSTEM ANALYSIS
FOR THE
HUNTSVILLE OPERATIONAL SUPPORT CENTER
DISTRIBUTED COMPUTER SYSTEMS

SUMMARY

The Huntsville Operations Support Center (HOSC) is a distributed computer system used to provide real time data acquisition, analysis and display during NASA space missions and to perform simulation and study activities during non-mission times. The primary purpose of this research is to provide a HOSC system simulation model that may be used to investigate the effects of various HOSC system configurations. Such a model would be valuable in planning the future growth of HOSC and in ascertaining the effects of data rate variations, update table broadcasting and smart display terminal data requirements on the HOSC HYPER channel network system.

A simulation model was developed and programmed in three languages BASIC, PASCAL, and SLAM. Two of the programs are included in this report, the BASIC and the PASCAL language programs. SLAM is not supported by NASA/MSFC facilities and hence was not included. The statistical comparison of simulations of the same HOSC system configurations are in good agreement and are in agreement with the operational statistics of HOSC that were obtained.

Three variations of the most recent HOSC configuration have been run and some conclusions drawn as to the system performance

under these variations. Section 3.4 discusses these results and conclusions.

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

1.0 INTRODUCTION

1.1 HOSC System Overview

Marshall Space Flight Center (MSFC), Huntsville, Alabama, has implemented the Huntsville Operations Support Center (HOSC) to provide real time data acquisition, analysis, and display during NASA space missions. The HOSC is a distributed computer system composed of a network of large minicomputers and various peripheral equipment. Primarily designed to provide support for the Space Shuttle, Space Telescope, and Space Laboratory missions, the HOSC has the inherent flexibility to be expanded to meet the needs of future missions as well as providing MSFC with a large computer resource that can be used to support several non-mission activities.

The HOSC facility has been structured to include five large minicomputers and various peripheral equipment. The current network computers are each semi-dedicated to specific mission tasks (e.g. Space Shuttle Main Engine Data Analysis) and include three Perkin Elmer 3244 computers, a Perkin Elmer 8/32c computer, two DEC VAX 11/780 computers and a DEC 11/24 computer. An important role of the Perkin Elmer computers is acting as real time data receivers for mission data arriving via satellite and direct ground links from the Kennedy Space Center Firing Room at Cape Canaveral FL. These computers also act as a gateway to the network for the data which is needed by other mission activities supported by the other computers and peripherals. Peripheral equipment in the system includes two twelve channel Genisco Digital Television (D/TV's), strip recorders, and numerous unintelligent data terminals.

Foreseeable future expansion will include at least five more minicomputers, many more D/TV displays (possibly to 50), several more strip recorders, and intelligent data terminals.

The HOSC currently provides support for MSFC non-mission activities such as the Total POCC Preplanning Activity with future expansion providing data management resources for other non-mission activities. These activities might include the DEC IGDS (interactive graphics) and XEROX SIGMA (text processing) operations. All of these activities would be permitted use of the network resources through the Network Systems Corporation HYPER channel broadband local area network.

1.2 Scope of Report

In order to achieve the flexibility and efficiency needed by the HOSC, an analysis of the present system has been performed. This analysis coupled with projected system growth will insure that the HOSC remains a viable computing resource for MSFC. This report contains a summary of the baseline data gathered to begin the analysis of the HOSC computer network, Section 2.0, results of the analysis, Section 3.0, and a literature/bibliography Section 4.0. The report describes in detail some of the network components and also makes first iteration recommendations concerning network operations. This document should not be considered an end item since work still remains to be done in completely characterizing all the subtleties of the HOSC system.

1.3 Conclusion

From the work done thus far in the program, several conclusions and recommendations can be made.

A. Proposed IGDS/SIGMA Interface With HOSC

Network Systems Corporation does not currently produce hardware for the HYPER channel to XEROX processor interface. Consequently, a great amount of effort would be required to interface the SIGMA system directly with the HOSC HYPER channel.[1] A possible solution might be to interface the XEROX SIGMA to the network through a HYPER channel supported processor such as another DEC VAX. Feasibility of the VAX/XEROX interface has not been explored and may also present problems. A definite possibility to solve this problem is to develop a suitable software/hardware approach.

The DEC IGDS system interfaces with the HYPER channel and will present no obvious problems since the PDP-11 processor interface adapters are currently marketed by Network Systems Corporation.

B. CSO/HOSC Link Via HOSC HYPER Channel Adapter 4 For OI Data Exchange

The current plan is to interface CSO with HOSC using a separate trunk of adapter 4. By connecting the two installations with a separate trunk, CSO will be disallowed easy and immediate access to the HOSC resources on the HYPER channel. Because of the HYPER channel adapter design, direct trunk to trunk exchange of data is not possible. For trunk to trunk transfers, data from the initiating trunk must be channeled through a processor on the common adapter and retransmitted by the processor over the other trunk.[2] It however, it is desireable to prevent CSO from easy access to the total HOSC resources, then the use of separate trunk is a good approach.

C. Summary of Analysis Activity

Progress on the analysis of HOSC has so far been steady but somewhat slow due to the difficulty in obtaining some needed baseline data. Below is a summary of the documentation accumulated to conduct the analysis effort.

- Perkin Elmer Corporation

  3240 User's Mannual 29-685

  3240 Memory System Manual 29-688

  8/32 User's Manual 29-394

  8/32 Memory System Manual 29-428

  (These manuals must contain the actual HOSC Computer internal DMA to I/O setup.)

- Network Systems Corporation

  PI40 Peripheral Interface Manual

  (Perkin Elmer I/O Bus Interface)

  PI10 PI Manual or PI11 PI Manual

  (Dependent on configuration of PDP-11 IGDS system: PI10 for DR11-B general purpose direct memory access or PI11 for DR70 MASSBUS interface.)

  NETEX Software Documentation

- Marshall Space Flight Center

  Completed system computer data rate flows.

D. Effects of Data File Dumps

It is desired to make large data file transfers on a periodic basis to refresh the data display terminals data base. This type of activity can create a log/jam effect on the most active data sources if the number of data bytes to be transferred are large enough to create waiting times. The basic relationship involves a tradeoff between the amount of storage of data by the data sources, their rate of data accumulation and the time required to transfer the data files.

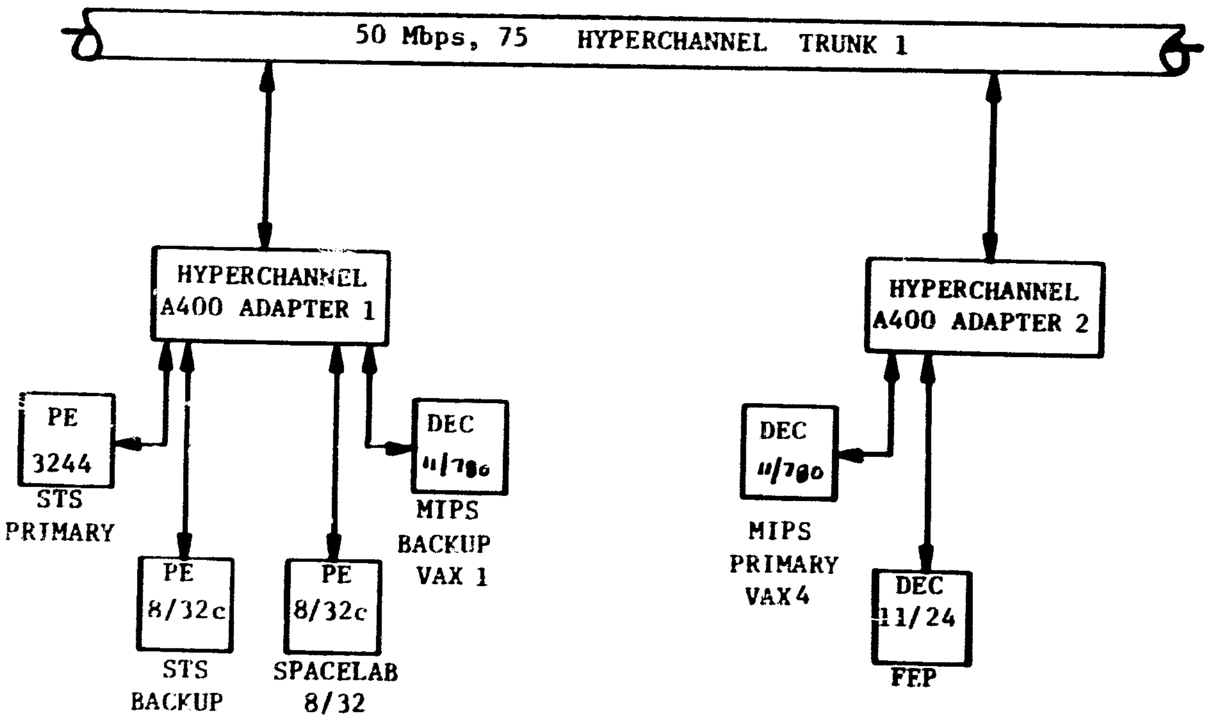
This problem is discussed in Section 3.3 and 3.4 in detail.

2.0 HOSC SYSTEM DETAILS

The primary purpose of the Huntsville Operation Support Center is providing MSFC engineers with a near real time summary of vital information describing the operational status of certain components of the Space Shuttle during pre-launch and launch activities. This information allows MSFC engineers and contractor personnel to act in a support capacity to mission personnel at Kennedy Space Center (KSC), Cape Canaveral, Florida, and also Johnson Space Center (JSC), Houston, TX. MSFC support is provided by teams responsible for the Space Shuttle Main Engine (SSME), External Tanks (ET), Solid Rocket Boosters (SRB), Main Propulsion System (MPS) and the Range Safety System (RSS). Additional mission support is provided for various mission activities and programs that are the responsibility of MSFC personnel.

> During powered flight, the HOSC will receive only data which is in the LPS (Launch Processing System) at KSC. The Shuttle support team will be in the HOSC during this phase of the mission and will be the point of contact with the JSC Mission Evaluation Room (MER) for problem discussion and resolution as required and will be on call during orbital operations. The Space Lab and experiment support team will be located in the HOSC during orbital operations when applicable
>
> Following completion of the active Shuttle vehicle support activities, data is recalled as required for more detailed analysis, and initial preparation is made to provide support to postflight evaluation.[3]

The HOSC is located in the west end of A-wing, building 4663 on Martin, Road, MSFC. Figures 1 and 2 show the functional components of the HOSC system and gives each component a referencing number that will be used in describing the system activities.



ORIGINAL PAGE IS OF POOR QUALITY

Figure 1. Original HOSC HYPER Channel Network Configuration

ORIGINAL PAGE IS
OF POOR QUALITY

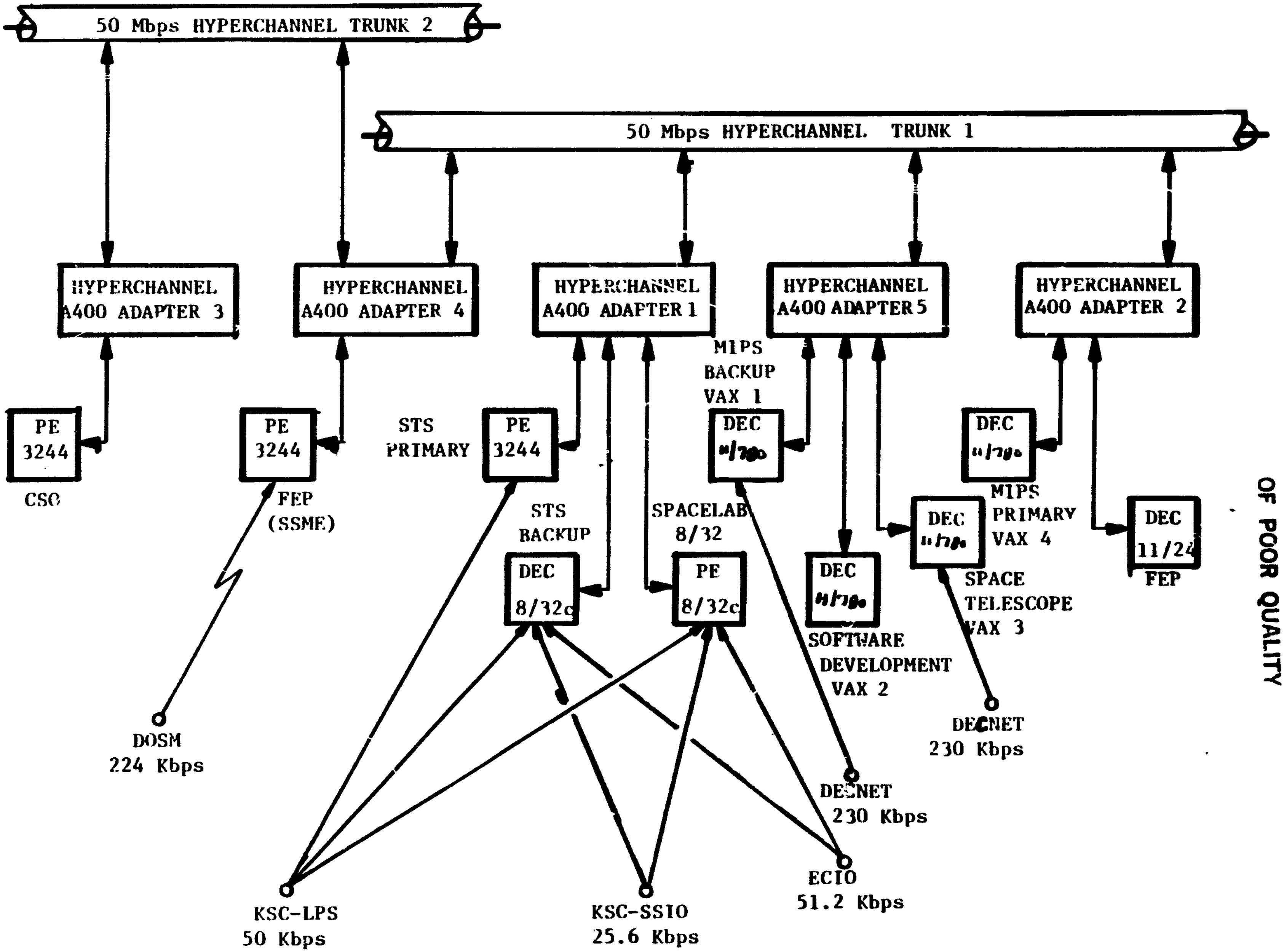


Figure 2. Proposed HYPER Channel Network Configuration

ORIGINAL PAGE IS
OF POOR QUALITY

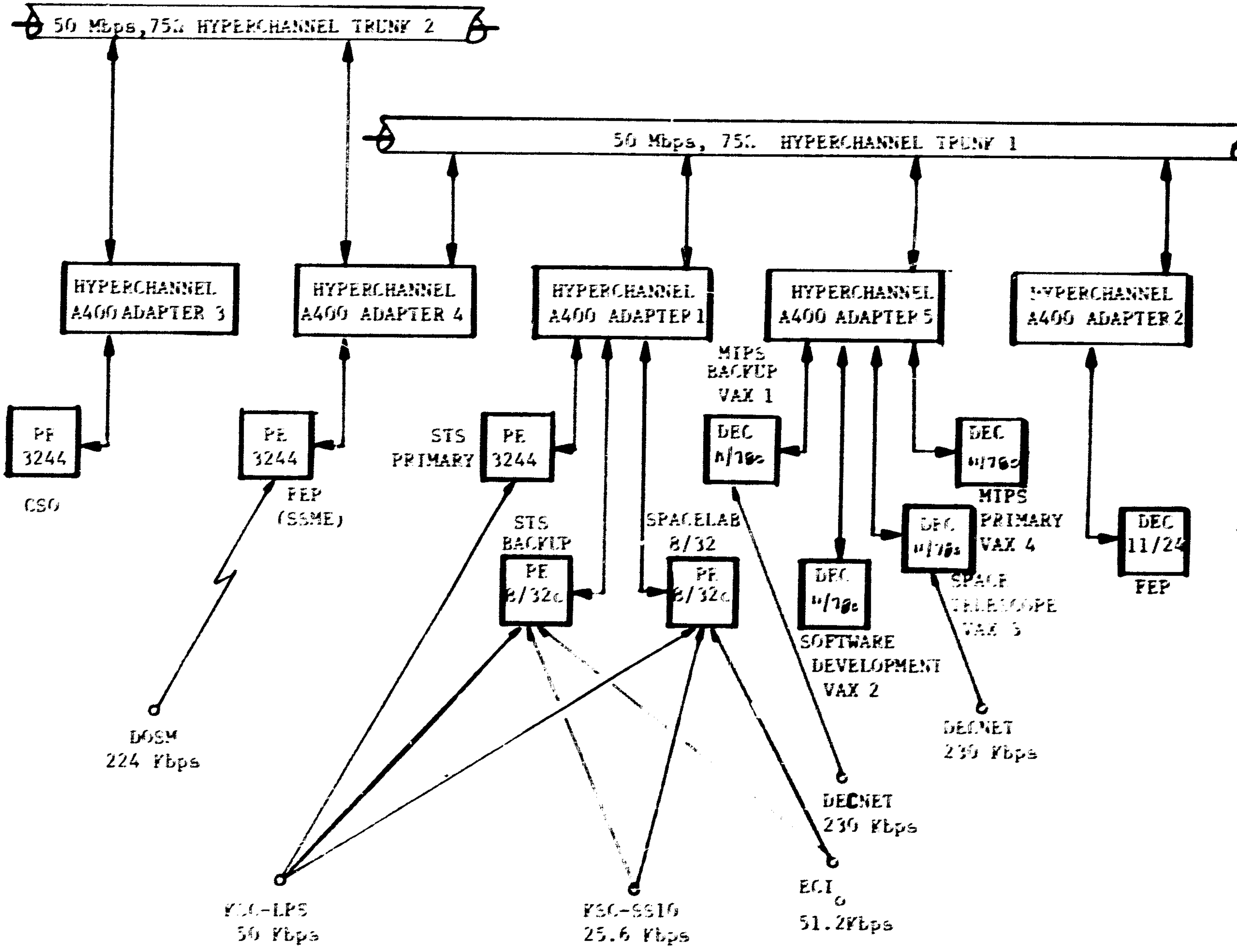


Figure 2a. Proposed HYPER Channel Network Configuration Rev 1.
(MIPS Primary VAX 4 Connected to Adapter 5)

2.1 HOSC System Activities

In addition to the mission activities, the HOSC also provides support to several non-mission activities at MSFC. Details of all the HOSC activities are described below and summarized in Table 1.

2.1.1 Total POCC Preplanning

The POCC activity is an ongoing simulation activity for which the HOSC lends computer resources. This activity is in no way keyed to the real time mission activities and must be viewed as a con tinuous daily activity.

The POCC activity's impact on the HYPER channel network is basically that of continuous data transfers between the MIPS Primary Computer (VAX4, A400 Adapter 2) and the MIPS Backup Computer (VAX1, A400 Adapter 5). During each 24 hour period 150,000 512-byte blocks of data are transferred. Six times each day, an 8344 byte block is transferred (50,000 bytes cummulative). The remaining 100,000 512-byte blocks are transmitted randomly, but on an evenly distributed basis, throughout the day.

2.1.2 ECIO Data Stream

The POCC activity generates a continual 51.2 kilobit/second data stream known as the Experimental Computer Input/Output (ECIO) data stream. This data stream is ongoing and concurrent with the POCC activity. Data is transferred from MIPS Backup (VAX1, A400 Adapter 5) to the Spacelab 8/32 (PE 8/32c, A400 Adapter 1).

TABLE 1. HOSC DATA TRANSFERS

I. ROUTINE DAILY ACTIVITIES (Launch Independent)

A. Total POCC Preplanning Activity

Resources involved: MIPS Primary (VAX4, Adapter 2)
MIPS Backup (VAX1, Adapter 5)

Quantity of data: 150,000 512-byte blocks daily

B. ECIO Data Stream (Generated by POCC)

Resources involved: MIPS Backup (VAX1, Adapter 5)
Spacelab 8/32 (PE 8/32c, Adapter 1)

Quantity of data: 51.2 K bits/second concurrent with POCC.

C. IGDS/SIGMA Activity (Proposed)

Resources Involved: DEC IGDS and XEROX SIGMA and
communication with other resources
as needed.

Quantity of data: TBD

II. LAUNCH DAY ACTIVITIES:

A. Routine Daily Activities (See Above)

B. Main Engine Data

Resources Involved: STS Primary (PE 3244, Adapter 1)
MIPS Backup (VAX1, Adapter 5)

Quantity of Data: 50 K bit/second stream (T-8 hours to T+12 minutes)

C. OI Data Stream

Resources Involved: FEB SSME (PE 3244, Adapter 4)
CSO Computers (Adapter 4)
STS Primary (PE 3244, Adapter 1)
MIPS Backup (VAX1, Adapter 5)

Quantity of Data: 128 K bit/second (T-9 sec to T+12 minutes)
into FEP and then to CSO. 40% will also
be transferred to STS and MIPS.

TABLE 1. HOSC DATA TRANSFERS (Continued)

D. Engineering Display Changes

Resources Involved:	STS Primary (PE 3244, Adapter 1) STS Backup (PE 8/32 Adapter 1) Spacelab 8/32 (PE 8/32, Adapter 1)
Quantity of Data:	Insignificant

2.1.3 Main Engine Data

Space Shuttle Main Engine data is collected and dissiminated at the HOSC during a launch day activity only. Data is funneled through the HYPER channel network to MIPS Backup (VAX1, A400 Adapter 5) via STS Primary (PE 3244, A400 Adapter 1). STS Primary accepts a continual 50 kilobit per second data stream directly from the KSC firing room from 9 seconds before launch to 12 minutes after launch (MECO). Approximately 24 percent of this 50 Kb/s stream (12 Kb/s) is transferred over the network to MIPS Backup.

2.1.4 OI Data Stream

The OI data stream is a 128 kilobit per second data stream arriving at FEP SSME (PE 3244, A400 Adapter 4) on launch day only (t-9 seconds to T+12 minutes). This data will have a much greater future impact on the network than it does currently. The SSME computer acts as a front end processor for accepting this data stream from Goddard Space Flight Center and then writes the received data directly onto a magnetic tape for later transport to CSO. Later in the program this data will be shipped in its entirety over a separate HYPER channel trunk attached to A400 Adapter 4 to CSO. Additionally, about 40 percent of the data stream will be shipped over the HOSC network to supply and supplant the data currently being transferred by the Main Engine Data Activity.

2.1.5 Engineering Display Changes

This activity adds almost insignificantly to the total HYPER channel trunk traffic. The activity involves a transfer from STS Primary to STS Backup and the Spacelab 8/32 (PE3244 to two PE8/32's, A400 Adapter, 1 only) of the name of each engineering console display format that is changed during the pre-launch and launch activities (T-9 seconds to T+12 minutes). This activity will be ignored in the HOSC system analysis due to its negligible contribution to total HYPER channel system traffic.

2.1.6 Proposed Activities

The most immediate proposed expansion of the HOSC network would allow two other non-mission activities access to the resources of the HOSC network. This activity would specify an additional A400 Adapter to allow resource sharing with the XEROX SIGMA system and the DEC PDP-11 IDGS system. Direct interface with the A400 is available for the PDP-11 but not for the XEROX system. A possible solution to allow the XEROX system access to the network through the A400 might be to use a compatible computer such as a DEC VAX 11/780 as a front end processor for the XEROX system. This activity is incompletely specified and will not affect the immediate analysis of the HOSC system.

2.2 HOSC System Components

The heart of the HOSC system is the Network Systems Corporation HYPER channel. The HYPER channel is a high speed digital communications facility that is used for interconnection of computer resources in a computing installation. The following sections describe the computer resources of the HOSC and how they are interconnected using the HYPER channel.

2.2.1 Computer Resources

2.2.1.1 DEC VAX 11/780

The HOSC makes use of Digital Equipment Corporation's VAX 11/780 computer as computational devices. The system currently includes two VAX computers (VAX1 and VAX4) designated as MIPS Primary and MIPS Backup. Future expansion will add two other VAX computers (VAX2 and VAX3) designated as Software Development and Space Telescope.

VAX computers support a 32-bit work architecture that is designed to aid in system throughput. Data transfers are accomplished via a 32-bit high speed data structure. This structure ties together the central processor, main memory, the UNIBUS subsystem, the MASS BUS subsystem and the DR780 high speed direct memory access subsystem. The 32-bit word architecture of the VAX establishes a virtual address space of 4.3 billion bytes of user addressable memory. A conceptual diagram of the VAX 11/780 bus structure is shown in Figures 3 and 4.

The Synchronous Backplane Interface (SBI) is the data path that links the central processor, the memory subsystem and the hardware adapters provided for the UNIBUS and MASSBUS. When interfaced to the SBI, the memory subsystem, the central processor, and the I/O controllers are known as NEXUSs.

All NEXUSs receive every SBI transfer. Logic in each NEXUS determines whether the NEXUS is the designated receiver for this transfer. Data transfers can occur from

CPU to memory subsystem

I/O controller to memory subsystem

CPU to I/O controllers.

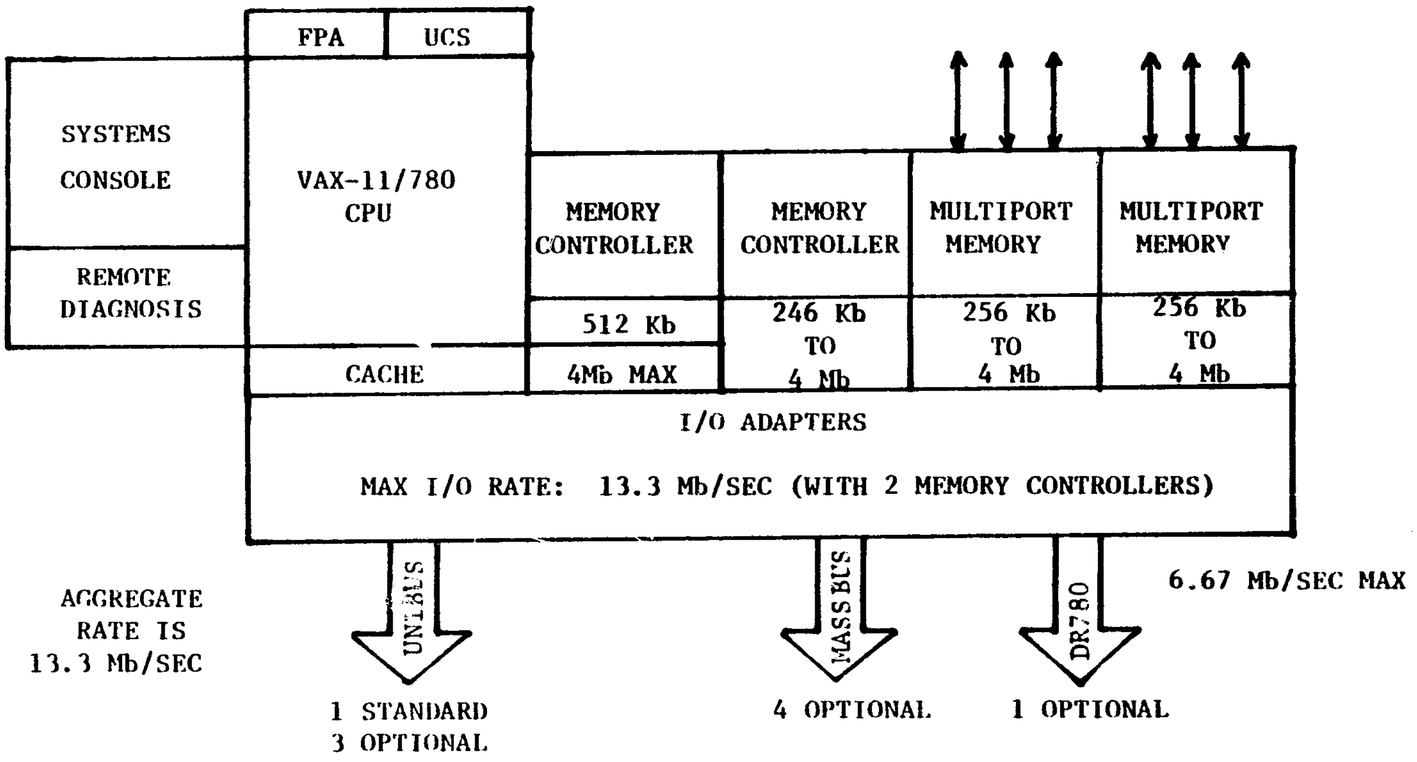


Figure 3. Block Diagram of VAX 11/780 Computer

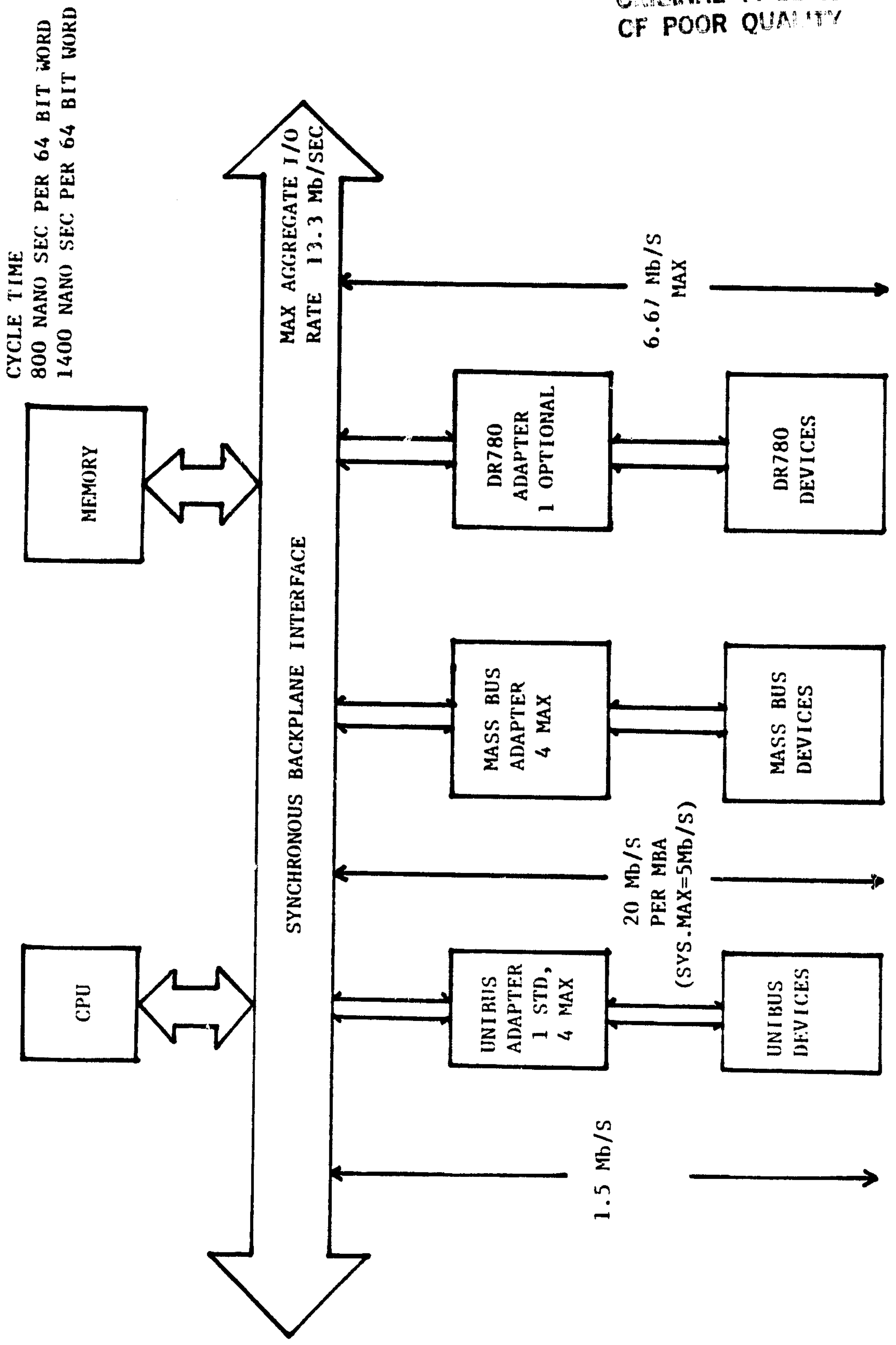


Figure 4. Basic Bus Configuration of VAX 11/780

The maximum, aggregate data transfer rate on the SBI is 13.3 megabytes per seconds which can be derived from the following information.

- 200 Nanoseconds/cycle = 5 million cycles/second
- Each cycle can carry an address (memory request) or for byte of data
- Thus, one cycle is used to request eight bytes of data (to be read or written), and two cycles are used to carry data (at four bytes/cycle).
- Five million cycles/second * 4 bytes/cycle = 20 million bytes/second
- 20 * 2/3 (1 of every 3 cycles is an address) = 13.3 million byte/second.[4]

The memory controller is the NEXUS used to interface the memory subsystem to the SBI. A system may have more than one memory controller as in the case of a two controller interleaved memory configuration.

The UNIBUS (UBUS) is a high speed asynchronous data system that allows communication between peripheral hardware and the VAX 11/780. The VAX 11/780 is capable of supporting 4 UBUS subsystems; one is standard with three more optional. The UBUS is connected to the SBI through a UBUS adapter (UBA) which performs priority arbitration among the devices on the UBUS. The primary functions of the UBA are to provide:

(1) Access to UBUS address space from the SBI

(2) Mapping of UBUS address to SBI addresses for UBUS direct memory access (DMA) transfers to system memory.

(3) Data transfer paths for UNIBUS device access to random SBI memory addresses and high speed transfer for devices that transfer to consecutive increasing address.

(4) UNIBUS interrupt fielding

(5) UNIBUS priority arbitration.

All of these services are completely transparent to UBUS users.

The address mapping function is necessary because the UBUS has only 18 data lines thus providing an apparent memory addressing capability of 2^{18} or 200 kilobytes. The UBA, however, provides the capability of mapping the UNIBUS addresses into SBI addresses so that the full memory of the system can be accessed. (Full system memory is 16 array boards of 256 kilobytes each for a total of 4 megabytes.)

The UBA accepts either of two forms of input from the UBUS: hardware generated interrupt or 'irect memory access transfer. Each device connected to the UBUS uses one of five priority levels for requesting bus service. The NonProcessor Request (NPR) is used when the device requests a direct access transfer to memory or some other device and does not require processor intervention. A Bus Request (BR) is used when the device wishes to interrupt the BPU for service. Such service might be a CPU directed data transfer or the informing of some error condition that exists at the perepheral. The NPR has the highest priority with four levels of BR following (BR7-BR4).

Since there are only five priority levels and more than one device may be connected to a specific request level, if more than one device makes the same request, the device that is electrically closest to the UBS receives higher priority.

The Non Processor Request for direct memory access is a very important feature of the UBUS subsystem. These DMA transfers can be divided into two groups: random access of noncontiguous addresses and sequential access of sequentially increasing address. For random access, each UBUS transfer is made through the Direct Data Path (DDP, one per UNIBUS) and is mapped into an SBI transfer. This procedure allows only one word of data to be transferred during a single SBI cycle. For devices capable of requesting sequential access services, use is made of Buffered Data Path (BDP). Each UNIBUS provides 15 such BDPs. The BDP stores the data so that four UBUS transfers are performed for each SBI transfer.

The DDP must be used by devices not transferring to consecutive increasing addresses or by devices that mix read and write functions. The maximum throughput via the DDP is about 425 kilo words per second for write operations and 316 kilo words per second for each read operation. These rates will decrease as other SBI activity increases.[4]

Maximum published throughput via the BDP is about 695 kilo words per second for both read and write operations but actual expected throughput rates are only 1.5 mega bits per second. This rate will also decrease as other SBI activity increases.[1,5] BDP transfers are restricted to block transfers where a block is defined as equal to or greater than one byte. All transfers within the block must be to consecutive and increasing addresses and all transfers must be of the same function type (Read or Write).

The MASSBUS subsystem and the DR780 high perforfanc 32-bit parallel interface will not be described in this report since an understanding of their functional characteristics is not needed to determine their relative impacts on the HYPER channel network. The influence of both may be felt indirectly, however, since activity on the MASSBUS or DR780 will translate to SBI activity which will affect DDP and BDP transfer rates as described perviously.[4]

Likewise, the VAX CPU will not be described in detail but several comments may be made about the CPU's effects on throughput. The CPU represents the most intensive traffic load on the memory subsystem and hence on the SBI. Obviously if the processor is engaged in computing, it will request data much more often than it will write data. Fortunately the large memory cache (8 kilo bytes) available to the CPU reduces the SBI traffic load considerably.

In terms of the SBI traffic, impact on the processor's speed, published figures[4] indicate that in a system with two memory controllers, the processor will be slowed about four percent per averaged megabyte per second of I/O traffic. The impact of a single memory controller is to slow the processor by a factor varying from two to four. Table 2 summarizes the DEC VAX 11/780 I/O characteristics.

TABLE 2. SUMMARY OF DEC VAX 11/780 DATA I/O CHARACTERISTICS

PROCESSOR :	32 bit words
MAIN MEMORY	
Virtual Address Space :	4.3 billion bytes
Cycle Time :	800 nanoseconds per 64-bit read 1400 nanoseconds per 64-bit write
I/O UNIBUS Adapter	
Maximum UNIBUS I/O Rate:	1.5 Mb/sec through buffered data paths.
Buffered Data Path :	15 total, 8 byte buffer in each 695 K words/second for read operations 695 K words/second for write operations Used for fast DMA transfers
*Direct Data Path :	425 K words/second for write operation 316 K words/second for read operation Used for transfers to non-consecutive memory locations.

All data rates subject to degradation as traffic on SBI increases. (SOI allows communication interfaces between CPU, Memory, UNIBUS and MASSBUS.)

** Maximum aggregate throughputs on UNIBUS is only 1.5 Megabytes/second.

2.2.1.2 Perkin Elmer 3244

The Perkin Elmer (PE) 3240 series computer is a high throughput machine with a 32 bit architecture. The HOSC currently uses two PE 3244 machines with primary responsibilities as front end processors (FEP) receiving real time data streams from the KSC firing room.

A block diagram of the 3240 model computer is shown in Figure 5. Detailed information regarding the 3244 has not been obtained but a brief description of the 3244 architectures follows.

The 3244 memory subsystem is organized into banks each capable of handling 4 megabytes of addressable memory. Total system memory ranges from 256 kilo bytes in one bank to a full system complement of four 4 megabyte banks for a maximum of 16 megabytes of addressable memory. All memory is connected to a common memory bus which consists of two undirectional, asynchronuous, 32 bit busses. One bus is dedicated to memory write functions and the other is dedicated to memory read functions.

Input/Output is accomplished by up to five external communication busses: one multiplexer bus for medium speed devices and up to four high speed Direct Memory Access (DMA) busses. Each DMA bus supports eight high speed bidirectional ports. Each DMA port is controlled by a selector channel that controls and terminates transfers through the CPU. This selector channel is controlled through the multiplexer bus. Once the channel is activated, the processor is released and is free to continue processing. Published I/O transfer rates for the PE 3244 DMA bus indicate that transfer

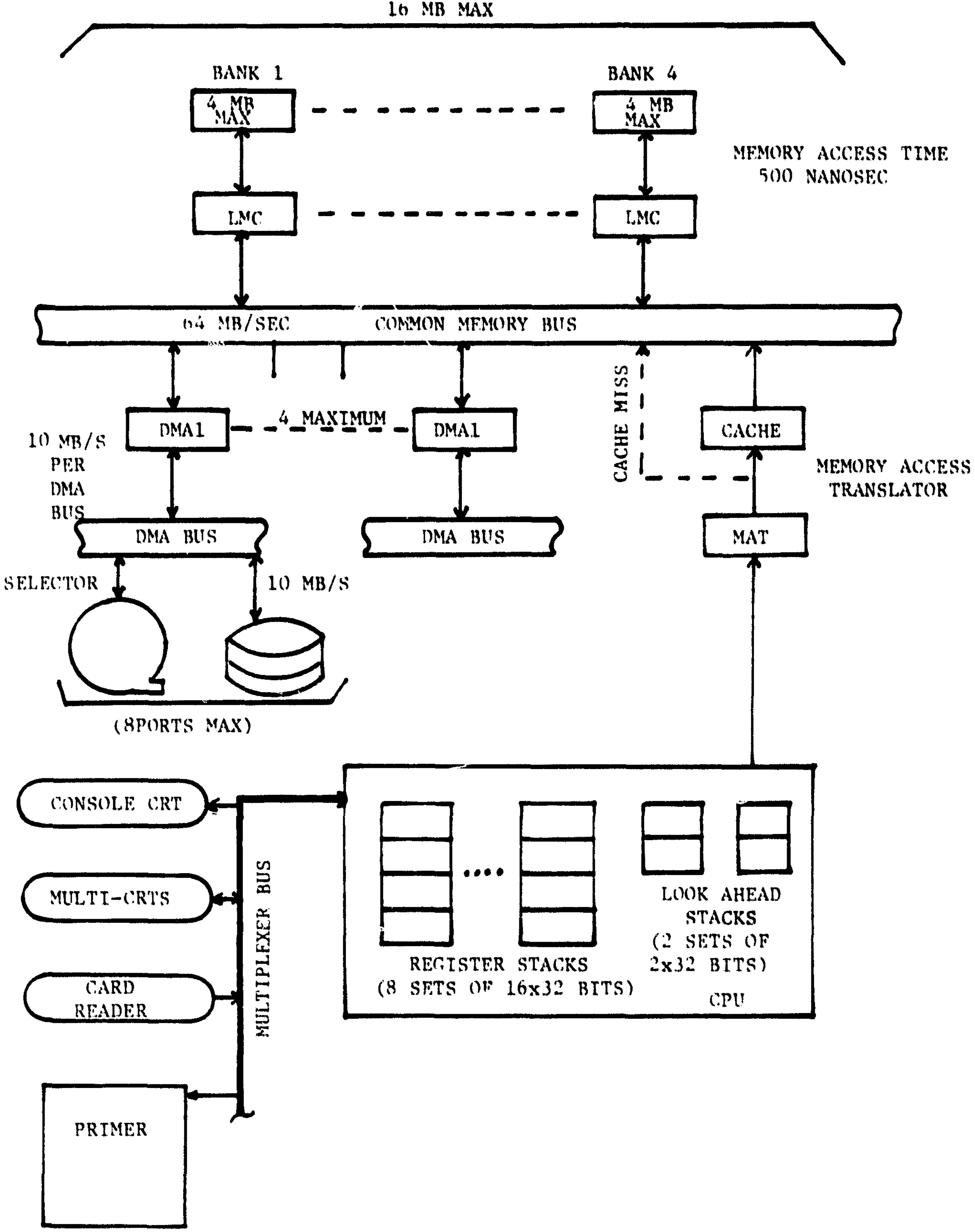


Figure 5. Block Diagram of PE 3244 Computer

rates of up to 10 megabytes per second burst mode are possible for each DMA bus.[6] Table 3 summarizes the PE/3244 I/O characteristics.

2.2.1.3 Perkin Elmer 8/32c

Detailed information about the 8/32 computer has not been obtained, but conceptually, the 8/32 is a machine similar in architecture to the 3244. A significant difference is that the 8/32 is capable of supporting only one DMA bus. This DMA operates in a burst mode capable of transferring 6 megabytes per second. The 8/32 will allow configuration with a buffered selector channel that accomplish the 6 MB/s rate by transferring the data in 14 half-word blocks.[7] Table 4 summarizes the estimated PE 8/32c I/O characteristics.

2.2.2 NSC HYPER channel

The Network System Corporation HYPER channel (HC) is a broadband local area communication network supporting data transmissions between network users at a rate of 50 megabyte per second. The HYPER channel network (HCN) serves to interface and interconnect various sizes of mainframe computers of differing manufacturers (e.g., UNIVAC, DEC, CRAY, PERKIN ELMER) with other peripherals such as data entry terminal card readers, printers, mass storage devices and other networks. Communication is provided over a passive 75 ohm coaxial cable called a trunk.

TABLE 3. SUMMARY OF PERKIN ELMER 3244 I/O CHARACTERISTICS

PROCESSOR :	32 bit/word
MAIN MEMORY	
Virtual Address Space:	4 Megabytes
Basic Memory Access Time:	500 nanoseconds
DMA BUS DATA TRANSFER RATE:	10 Megabytes/second-burst mode Maximum of 4 DMA busses can be supported.

TABLE 4. SUMMARY OF PERKIN ELMER 8/32 I/O CHARACTERISTICS

PROCESSOR :	32 bit/word
DMA BUS DATA TRANSFER RATE:	6 Megabytes/second by transferring data in 14 half-word blocks. Only one DMA can be supported.

Host computers gain access to the network trunk through hardware interfaces called processor adapters; unintelligent peripherals through device adapters. Network to network connection are accomplished with a link adapter which supports not only communication with standard transmission lines but also with microwave frequency RF links. Each network adapter may be connected to as many as four separate trunks and provides the service of trunk selection, trunk access, establishment of adapter to adapter virtual circuits and also provides user-to-adapter protocols. Network adapters contend for trunk control using a Carrier Sense Multiple Access scheme with prioritized staggered delays.[9]

The heart of the network is the A400 Adapter. The A400 is a microcomputer controlled interface device that allows up to 4 minicomputers of the same or mixed manufacturer types to transmit and receive data over the HYPER channel network. (All four trunk port may be connected to four channels of the same minicomputer.) The A400 provides a buffered interface between the trunk and the adapter. Some of this buffer is used to provide parallel to serial data stream conversion for host to trunk transmissions and serial to parallel conversion for trunk to host transmissions.

Each A400 adapter is composed of

- a 16 bit microprocessor with 4906 words of read only memory.
- a storage section consisting of
  - 1024 8-bit bytes of control memory with odd parity
  - 4096 8-bit bytes of control buffer with odd parity

16 working registers

16 trunk registers

256 extension register

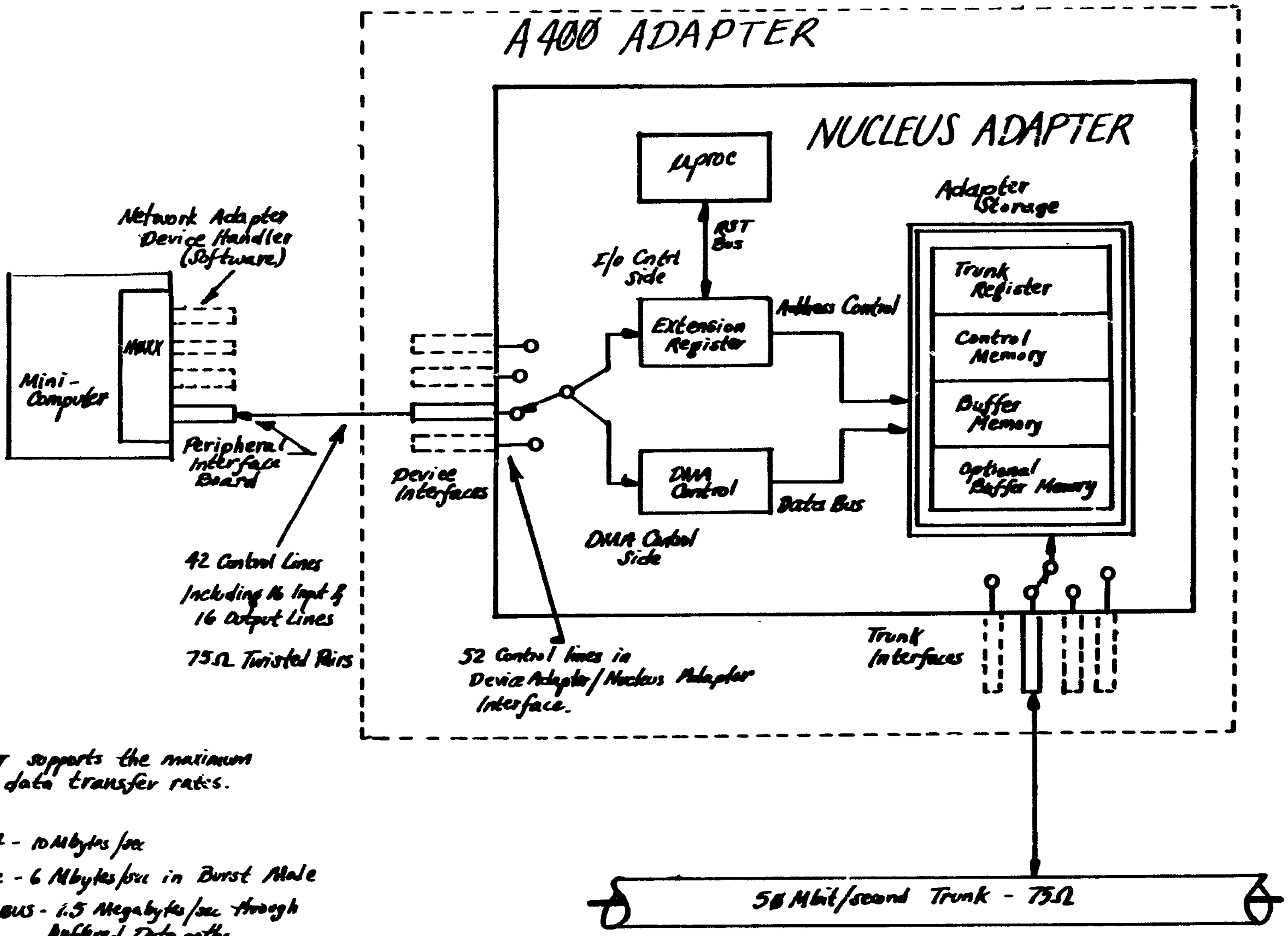
- one trunk interface.

The adapter can be expanded to contain

- 4 trunk interfaces
- 8192 8-bit bytes of buffer memory
- 1024 8 bit bytes of code conversion memory.

Additionally, the adapter has a peripheral device interface that provides a standard interface between the internal busses of the minicomputer and the A400. The peripheral interface adapter is separated from, but connected wich ribbon cables to, the nucleus adapter which provides the hardware resources such as the microprocessor and memory register.[8,10] (See Figure 6)

To perform an operation on the network, the minicomputer loads the necessary parameters into the internal registers on the interface and requests the adapter to perform the indicated functions. Whenever an adapter is not performing a function, it scans all attached ports for a request to perform a function. When a function request is detected, the adapter suspends scanning and initiates the execution of the function. The flow diagram of Figure 7 illustrates the handshaking between the A400 and host processor when data transfers are initiated. Notice that the host processor initiates all actions of the adapter. (A compilation of functions that can be accomplished by the A400 is illustrated in Table 5.)[8]



NOTES:

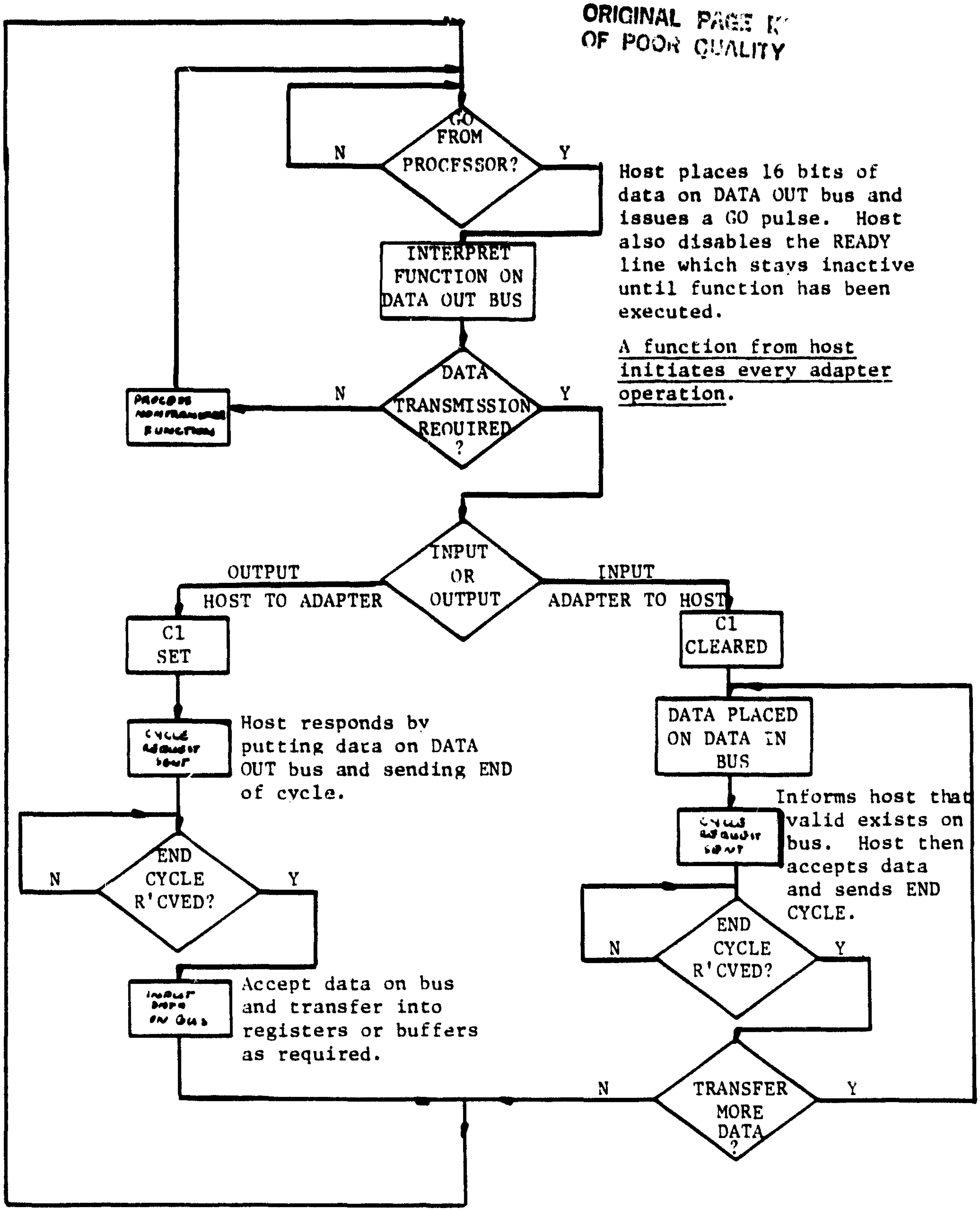
1. The A400 Adapter supports the maximum host processor data transfer rates.
   Typical Values:
   P/E 3244 - 10 Mbytes/sec
   PE 8/32c - 6 Mbytes/sec in Burst Mode
   DEC UNIBUS - 1.5 Megabytes/sec through buffered Data paths.

2. Microprocessor (logical) Interface consists of firmware (PROMS) to direct and control device transmissions, trunk selections, trunk tranmissions and Assembly/Disassembly.

3. Trunk side consists of I/o Buffering, Serial/Parallel Conversion, Checkword generator and checker, and Trunk Transceiver.

4. Device Interface Side consists of I/o Extension Registers and the DMA Controller.

Figure 6. NSC HYPERchannel Adapter block diagram.



Host places 16 bits of data on DATA OUT bus and issues a GO pulse. Host also disables the READY line which stays inactive until function has been executed.

A function from host initiates every adapter operation.

Host responds by putting data on DATA OUT bus and sending END of cycle.

Informs host that valid exists on bus. Host then accepts data and sends END CYCLE.

Accept data on bus and transfer into registers or buffers as required.

Figure 7. Host to A400 Adapter Data Exchange

TABLE 5. A400 ADAPTER FUNCTION DESCRIPTION

CODE	FUNCTION
04	Transmit message
08	Transmit data
0C	Transmit last data
10	Transmit local message
24	Input message
28	Input data
40	Status
50	Dump extension register
60	Mark down port 0
64	Mark down port 1
68	Mark down port 2
6C	Mark down port 3
70	Mark down port 0 and re-route messages
74	Mark down port 1 and re-route messages
78	Mark down port 2 and re-route messages
7C	Mark down port 3 and re-route messages
A0	Read statistics
A4	Read and clear statistics
C0	Set test
C4	Set address and length
C8	Write buffer
CC	Read buffer
E0	Clear adapter
E4	End operation
E6	Clear and wait for message
E8	Wait for message.

Data can be transferred from host to adapter in two different modes: direct memory access (DMA) and register mode. In the DMA mode, the adapter uses an alternating buffer scheme. The adapter accepts data from the device into buffer memory. When the buffer is half full, trunk transmission of that amount of data is initiated as, the other half of the buffer is being filled. This filling and sending is continued until all data has been transferred. All DMA transfers are through the extension registers and are initiated by the adapter microprocessor and controlled by the adapter hardware.

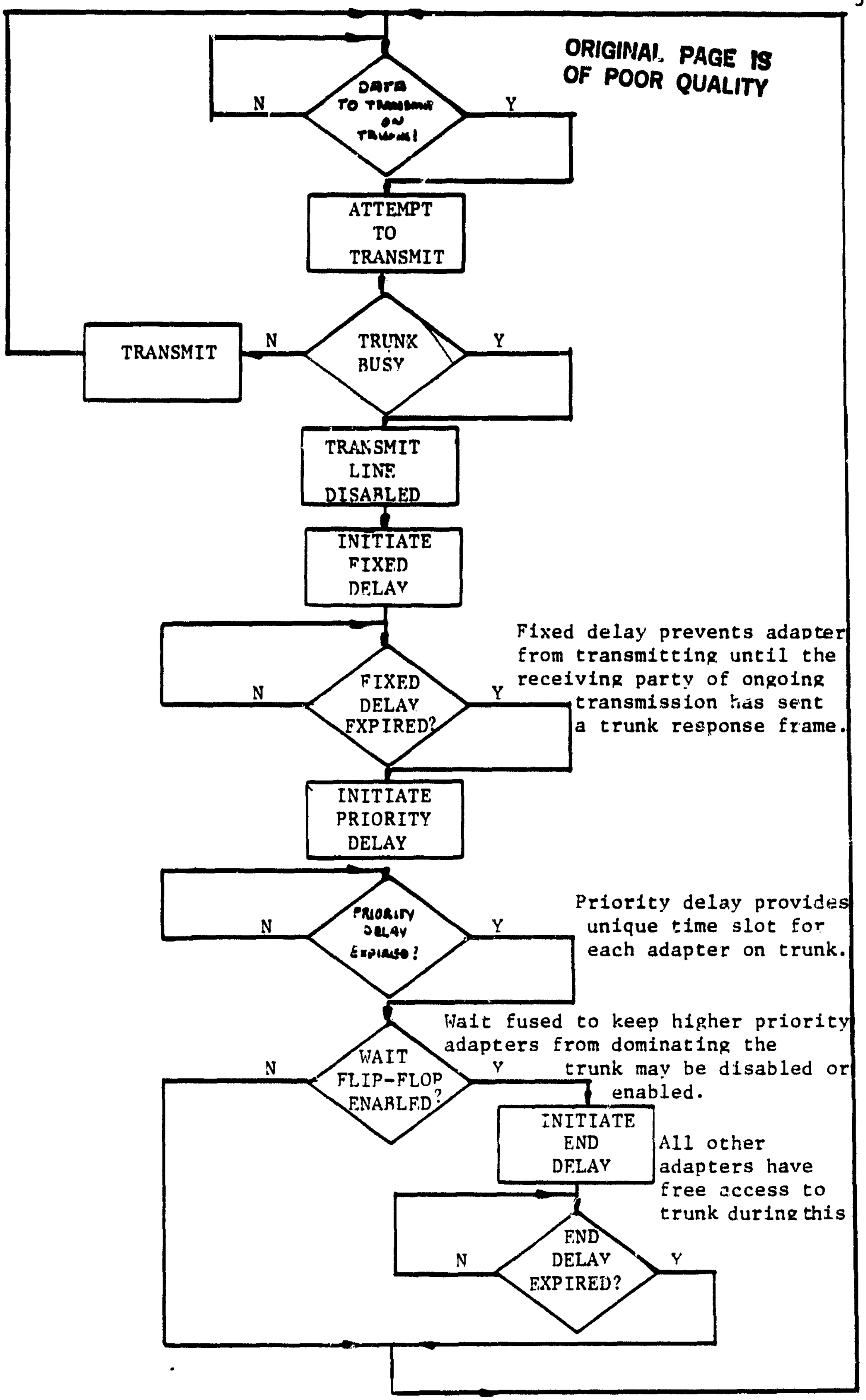
In the register mode, data movement is also between the device interface and the nucleus adapter but the DMA controls are not used. These data transfers are also through the extension register but initiated and controlled by the microprocessor.

Data transfers from adapter to adapter are accomplished by the trunk interface. The trunk interface consists of a passive coaxial cable that transmits data serially between two adapters. Each trunk can have up to 64 drops depending on the length of the trunk cable and its transmission qualities.

Transmissions on a trunk are initiated and monitored by the trunk driver which is a microcode program stored in the adapter PROMs. The extension register and the trunk registers support the PROM trunk driver. When an adapter is ready to transmit, it must first contend for use of the trunk. The method for contention is called contention allocation. It is so called because the trunk is allocated to an adapter based on the adapter's need to transmit.

The contention process can be summarized as follows. The adapter first just attempts to transmit on the trunk. If the trunk is busy, the transmitter is disabled. When the trunk becomes free, a fixed delay is initiated by the adapter. This prevents the adapter from transmitting until the receiving party of the most recent transmission has had time to receive a response frame. Upon expiration of the fixed delay, another delay called the priority delay is initiated. This delay is different for each adapter and provides a unique time slot for each adapter on the trunk. Annother delay, called end delay, is provided following the fixed delay. This delay is provided to insure that all adapters with higher priority have first access to the trunk. Obviously, with this trunk allocation scheme, higher priority adapters can dominate the trunk. To prevent this, each adapter has a flip flop in it that is known as the wait flip flop. This flip flop is set when the adapter transmits and is cleared when an end delay is signalled. This flip flop is intended to provide a more equitable contention environment. Although all adapters are equipped with wait flip flops, they may be disabled to provide assured trunk access.[9,10] Figure 8 shows the flow of the wait algorithm.

Upon gaining access to the trunk, either a function message or data can be transmitted in trunk frames. When a frame is transmitted all adapters receive the frame. The adapter compares the received adapter access code which is part of the frame header with its own code which can be set by thumblwheel switches inthe adapter. If and only if the codes match can the communication be accepted. (A zero in the receiving adapter code represents a "don't care" condition and



Fixed delay prevents adapter from transmitting until the receiving party of ongoing transmission has sent a trunk response frame.

Priority delay provides unique time slot for each adapter on trunk.

Wait fused to keep higher priority adapters from dominating the trunk may be disabled or enabled.

All other adapters have free access to trunk during this

Figure 8. Trunk Contention Delay Algorithm

the receiving adapter will accept any character in that code position.)

The receiving adapter responds to the receipt of a trunk transmission with a trunk response frame. This notifies the sending adapter of the status of the received message. Every transmission frame requires the receipt of a response frame or the sending adapter will time out and retry the transmission. This process will be repeated 256 times. If unsuccessful at transmitting the message, the adapter will terminate the operation and record some status bits for the host in the adapter extension registers.[10]

2.2.3 Other HOSC Components

In addition to the computer resources are several other devices in the HOSC. Currently these other devices act as peripherals to the processors on the HYPER channel network and consequently do not directly affect traffic on the network. Indirectly, they represent overhead processor activity and thus slow traffic throughput on the processor I/O busses. These devices will include a Gandalf solid state switcing matrix that acts to interface the MIPS consoles through VAX4. Also included in the peripherals are various strip recorders and three twelve-channel Genesco digital televisions that interface the engineering consoles through STS Primary (PE 3244), STS Backup (PE 8/32c) and Spacelab 8/32. No further descriptions of these devices are currently available.

3.0 HOSC ANALYSIS RESULTS

The HOSC system analysis initiated with a study of the system componente, the computers, the HYPER channel network, the data flow activity of each device and the input-output characteristics of each device. The system operation is statistical in nature and, although a mathematical analysis is possible, it is not feasible to make such an analysis with much fidelity. Rather a simulation model that emulates the HOSC system with good fidelity can be used to achieve information concerning average bus traffic, average waiting time, collision frequency and maximum waiting times. Furthermore, these parameters can be investigated as a function of HOSC system configuration, input-output variations, and data file dump requirements.

The development of a simulation model with good fidelity has been accomplished. The HOSC system has been modeled with three different program simulations and these three algorithms have been compared against each other. The purpose in using three algorithms was to insure validity of the simulation results, a necessity due to the lack of sufficient system statistics to validate a single simulation algorithm. The three algorithms are similar, but have been programmed in BASIC, PASCAL and SLAM.

BASIC is an engineering oriented language not at all unlike FORTRAN. This simulation program is the main program. The program is listed in Appendix I.

Although many simulation runs were made with simple system configurations that allowed the simulation algorithm to be verified, there is no need to present those in this report. The monthly reports document these earlier runs and the development of the algorithm. Rather it suffices to illustrate the simulation of the HOSC system as it is projected in configuration in Summer 1983.

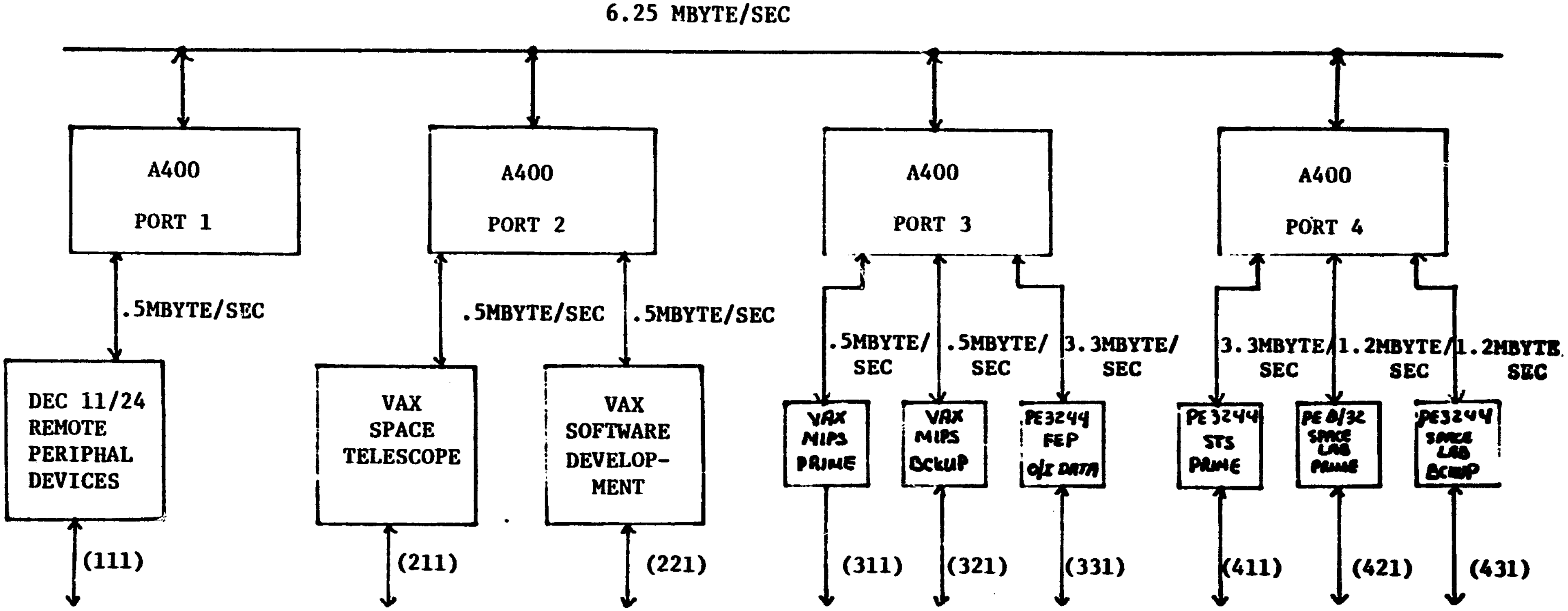
3.1 Typical Basic Algorithm Information Printouts

Figure 9 depicts the HOSC simulation configuration which is documented in this report. This configuration is perhaps more complex than the actual system configuration for the present, but it is the type of configuration that is desired in the near future. Not all devices are transmitters of data in this system configuration. The A400 labeled port 4 only receives data transmitted on the HYPER channel bus. Other devices receive outside data and transmit and received data over the HYPER channel bus. This system configuration was devised at a meeting between this investigator and NASA/MSFC HOSC personnel on March 9, 1983, and is typical of the configurations to be utilized for HOSC applications in the near future.

In order to determine the number of simulation runs necessary to produce representative statistics and to let the system algorithm achieve steady state, as would occur in the actual system, several runs with the same system configuration parameters but with varying numbers of data transfers were made and the statistics compared.

Figures 10 and 11 illustrate the program printout that depicts the system configuration of figure 9. The # of bytes accumulated refers to the number of bytes which a particular device will accumulate refers to the number of bytes which a particular device will accumulate from a source before it transmits that data to the approproiate destination. As may be noted in Figure 10, there is a

ORIGINAL PAGE IS OF POOR QUALITY



	(111)	(211)	(221)	(311)	(321)	(331)	(411)	(421)	(431)
DATA ARRIVAL RATE KBYTE/ SEC	15.0	4.0	6.4	1.0	1.0	.625	0	0	0
DATA TRANSFER BLOCK SIZE KBYTES/ TRANSMISSION	2.0	4.0	6.4	.512	.512	.625	2.0	2.0	2.0

SOURCE/DESTINATION	PROBABILITIES
111-311	.7
111-321	.3
211-431	1.0
221-421	1.0
311-321	.98
311-221	.01
311-211	.01
321-311	.98
321-221	.01
321-211	.01
331-421	1.0

DATA FILE DUMP	DATA BLOCK	TIME BETWEEN DUMPS
DEC11/24 to STS PRIME (PE3244)	64 KBYTES	12 SEC

Figure 9. HOSC Simulation Configuration

ORIGINAL PAGE IS
OF POOR QUALITY

```
# OF HYPERCHANNEL PORTS   =   4

# OF DEVICES FOR PORT 1   =   1
# OF DEVICES FOR PORT 2   =   2
# OF DEVICES FOR PORT 3   =   3
# OF DEVICES FOR PORT 4   =   3

# OF SOURCES FOR PORT  1  DEVICE  1  =  1
# OF SOURCES FOR PORT  2  DEVICE  1  =  1
# OF SOURCES FOR PORT  2  DEVICE  2  =  1
# OF SOURCES FOR PORT  3  DEVICE  1  =  1
# OF SOURCES FOR PORT  3  DEVICE  2  =  1
# OF SOURCES FOR PORT  3  DEVICE  3  =  1
# OF SOURCES FOR PORT  4  DEVICE  1  =  1
# OF SOURCES FOR PORT  4  DEVICE  2  =  1
# OF SOURCES FOR PORT  4  DEVICE  3  =  1

PORT,DEVICE.SOURCE          AVERAGE ARRIVAL RATE        # OF BYTES ACCUMULATED
 1     1     1                      15000                      2048
 2     1     1                      4000                       4000
 2     2     1                      6400                       6400
 3     1     1                      1000                       512
 3     2     1                      1000                       512
 3     3     1                      625                        625
 4     1     1                      0                          2000
 4     2     1                      0                          2000
 4     3     1                      0                          2000


LJ-LM DATA FILE DUMP     BYTES TO BE DUMPED     TIME BETWEEN DUMPS (IN SECONDS)
 1141                          64000                 12
```

Figure 10. System Configuration Parameters for 1333 Data Transfer Simulation.

```
WAIT TIME OF PORT  1  BEFORE TRANSMISSION REQUEST (IN EVENT OF A COLLISION)
 .000001
WAIT TIME OF PORT  2  BEFORE TRANSMISSION REQUEST (IN EVENT OF A COLLISION)
 .000002
WAIT TIME OF PORT  3  BEFORE TRANSMISSION REQUEST (IN EVENT OF A COLLISION)
 .000003
WAIT TIME OF PORT  4  BEFORE TRANSMISSION REQUEST (IN EVENT OF A COLLISION)
 .000004


CHANNEL SETUP AND RELEASE TIME =  .000025

CHANNEL DATA TRANSFER RATE =  6250000

PORT,DEVICE                  TRANSFER RATE          LOAD TIMES
 1       1                     500000                  .000002
 2       1                     500000                  .000002
 2       2                     500000                  .000002
 3       1                     500000                  .000002
 3       2                     500000                  .000002
 3       3                     3300000                3.0303030303E-7
 4       1                     3300000                3.0303030303E-7
 4       2                     1200000                8.33333333333E-7
 4       3                     1200000                8.33333333333E-7

COMBINATIONS (IJK - LMN)  PROBABILITY  TIME TO TRANSMIT Q(IJK) BYTES (IJK - LMN)
 111311                       .7         .004441
 111321                       .3         .004441
 211431                       1          .008345
 221421                       1          .013145
 311321                       .98        .001369
 311221                       .01        .001369
 311211                       .01        .001369
 321311                       .98        .001369
 321221                       .01        .001369
 321211                       .01        .001369
 331421                       1          8.65833333333E-4
```

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 11. System Configuration Parameters for 1333 Data Transfers.

data file dump of 64,000 bytes every 12 seconds as would be typical of a data file refresh operation.

The channel setup and release time is a parameter used to allow the HYPER channel to establish a transmission link and then to release the link after data transfer is complete.

As data is transferred across the system, each port vies for the bus in a contention scheme described in Section 2. Occasionally the ports will collide trying to transmit simultaneously. In Figures 12 and 13, a printout record of the results of a collision is illustrated. These printout records allow the operator to ensure the collision algorithm is working properly. As may be noted, 331 and 111 incurred a collision and 111 retransmitted first, since its assigned waiting time is less than 331.

Every time a data transfer occurs between two devices on the same port, the HYPER channel bus is not utilized and thus it is free for other transmissions except to the port involved in an interport data transfer. Figure 14 illustrates a printout record of an inter device data transfer. Figure 14 also illustrates a record of a data file dump.

The statistical printout of a simulation run is illustrated in Fugure 15. This run had 1333 data transfers and over three million bytes transferred.

3.2 Results of A Simulation Run

Inspection of Figure 15 will illustrate the information garnered by a simulation run. The items of interest are the bus busy time, collision frequency of each source, the average waiting time, the longest waiting time and the relative activity of each source.

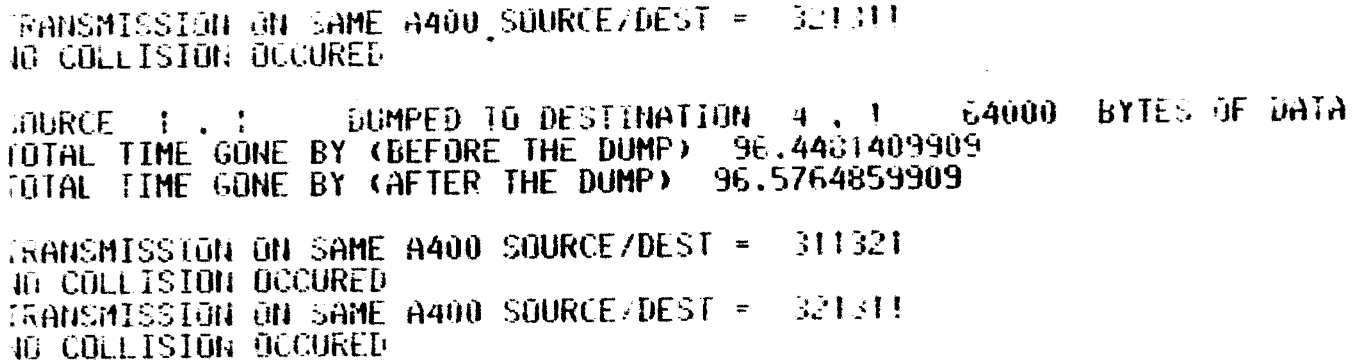
ORIGINAL PAGE IS
OF POOR QUALITY

```
A COLLISION OCCURRED BETWEEN  331  AND  111


ITERATION NUMBER  306

PROBABILITY  .7
 ANDOM NUMBER  .064597057222
 ROBABILITY  1

 ESTINATION  321

 ORT.DEVICE.SOURCE          TOTAL TIME GONE BY SINCE LAST TRANSMISSION
 1    1    1                  .135719166711
 2    1    1                  .798061499984
 2    2    1                  .668460499377
 3    1    1                  .301452499979
 3    2    1                  .301452499979
 3    3    1                  1.000001
 4    1    1                  999999
 4    2    1                  999999
 4    3    1                  999999

 ORT.DEVICE.SOURCE         REMAINING TIME TO TRANSMIT (NORMALIZED FASHION)
 1    1    1                  .000814166622
 2    1    1                  .201938500916
 2    2    1                  .331539500021
 3    1    1                  .210547500021
 3    2    1                  .210547500021
 3    3    1                 -.000001
 4    1    1                  999999
 4    2    1                  999999
 4    3    1                . 999999
COLLISION NUMBER  3
DATA TRANSFER  .004441
```

Figure 12. Typical Printout When Data Transfer Collision Occurs.

```
ORT,DEVICE,SOURCE       ACCUMULATED WAITING TIMES SINCE THE LAST TRANSMISSION
1     1     1                    0
2     1     1                    .302502499984
2     2     1                    .672901499977
3     1     1                    .305893499979
3     2     1                    .305893499979
3     3     1                    1.004442
4     1     1                    999999
4     2     1                    999999
4     3     1                    999999
ATA TRANSFER COMPLETED: TOTAL TIME GONE BY  28.340377331

ORT,DEVICE,SOURCE         REMAINING TIME TO TRANSMIT (NORMALIZED FASHION)
1     1     1                    .136533333333
2     1     1                    .197497500016
2     2     1                    .327098500023
3     1     1                    .206106500021
3     2     1                    .206106500021
3     3     1                   -.004442
4     1     1                    999999
4     2     1                    999999
4     3     1                    999999
EXT TRANSMISSION REQUEST BY  331
OTAL TIME THAT WILL BE GONE BY WHEN  331  STARTS TO TRANSMIT  28.340377331


TERATION NUMBER  387

ROBABILITY  1
ANDOM NUMBER  .608213251646

ESTINATION  421
```

Figure 13. Typical Printout When Data Transfer Collision Occurs (Continued).



```
TRANSMISSION ON SAME A400 SOURCE/DEST =  321311
NO COLLISION OCCURED

SOURCE  1 . 1     DUMPED TO DESTINATION  4 , 1    64000  BYTES OF DATA
TOTAL TIME GONE BY (BEFORE THE DUMP)  96.4481409909
TOTAL TIME GONE BY (AFTER THE DUMP)  96.5764859909

TRANSMISSION ON SAME A400 SOURCE/DEST =  311321
NO COLLISION OCCURED
TRANSMISSION ON SAME A400 SOURCE/DEST =  321311
NO COLLISION OCCURED
```

Figure 14. Typical Printout for Data File Dump and Inter Source Data Transfers.

```
OURCE TO DESTINATION          # OF TIMES TRANSMITTED
 111311                        481
 111321                        195
 211431                        95
 221421                        34
 311321                        183
 311221                        7
 311211                        6
 321311                        185
 321221                        1
 321211                        1
 331421                        95
OTAL                           1333

ORT.DEVICE.SOURCE      LONGEST WAITING TIME    AVG WAITING TIME      COLLISIONS
1    1    1             -.128345              -5.71128979232E-2       1
2    1    1             -.128345              -.04111054163           1
2    2    1             -.045081              -1.71633610833E-2       2
3    1    1             -.128345              -5.57095512472E-2       0
3    2    1             -.129714              -6.26356917362E-3       0
3    3    1             -.10163916665         -.03009355127           2
4    1    1              0                    0                       0
4    2    1              0                    0                       0
4    3    1              0                    0                       0
OTAL TIME GONE BY =  96.3994089909
US BUSY TIME =  6.1463851664
```

Percent Bus Busy Time = 6.34%

PORT,	DEVICE,	SOURCE	AVG. WAIT TIME %	LONGEST WAIT %
1	1	1	27.18%	94.00%
2	1	1	4.11%	12.84%
2	2	1	1.72%	4.51%
3	1	1	10.88%	25.07%
3	2	1	1.22%	25.35%
3	3	1	3.01%	10.17%

Number of Data Transfer Collisions = 6 Collisions

Number of Data File Dumps (11 to 41) = 8 (64,000 Bytes Each)

Bytes Transferred by Data File Dump = 512,000 Bytes

Bytes Transferred Source to Destination = 2,616,399 Bytes

Total Bytes Transferred = 3,128,399 Bytes

Figure 15. Statistical Summary of 1333 Data Transfer Simulation.

For the 1333 data transfer simulation with data file dump of 64,000 bytes every 12 seconds the parameters of interest are depected in the figure. Everything points to a satisfactory operation at this point with one notable exception, the longest waiting time. This Figure is -128.345 milliseconds (the negative sign indicates waiting time) for P.D.S. 111 which has an average time between transmission requests of 136.533 milliseconds (# of bytes accumulated divided by the average arrival rate of the bytes. The data is drawn from Figure 10.)

This waiting time amounts to 94% of the average time between transmission requests for P.D.S. 111 and serves a warning that P.D.S. 111 is on the verge of being overloaded. This may be alleviated by several means--changing the number of bytes to be accumulated before making a transmission request or by changing the data file dump time interval. In Section 3.4, this is discussed more fully.

3.3 Comparison of BASIC, PASCAL and SLAM Programs

Appendix II contains a listing of the PASCAL program and summary sheet for the HOSC simulation. The SLAM program listing is not included since NASA/MSFC does not carry SLAM software support. The three algorithms are compared in a broad sense in Table 6. The three algorithms were run for comparison purposes using 500 data transfers as the benchmark. The total bus time, bus utilization percentage, number of bytes transferred all compare very favorably. The number of collisions incurred vary due to differences in the collision algorithms used in the programs which were programmed by three different programmers as a check on the algorithms. The

TABLE 6

COMPARISONS OF BASIC PASCAL AND SLAM PROGRAMS

NO DATA FILE DUMPS SIMULATED

	BASIC	PASCAL	SLAM
Total Transmissions	500	500	500
Elapsed Time on Bus (Seconds)	36.08	36.44	35.24
Collisions	8	29	19
Waits	N.T.	83	36
Bus Utilization	5.82%	4.69%	6.02%
Bytes Transferred	979,795	987,360	985,939

N.T. = Not Tabulated

The algorithms programmed all have the following features:

1. Allows up to 9 A400 adaptors
2. Allows up to 9 computer devices per A400 adaptor
3. Allows up to 9 data sources per device
4. Allows each device to transfer large block of data on a periodic basis such as for CRT data base refresh
5. Allows assignment of individual waiting times to be assigned to each A400 in event of a collision
6. Allows the shortest assigned waiting time A400 to retry a transmission in the event of a collision
7. If a data transfer occurs between two devices on the same A400 (Inter A400 data transfer) it allows this to occur without tying up the bus
8. Allows for individual source data arrival rates
9. Allows for individual source data buffer sizes (relates to time between transmission requests)
10. Allows for individual device to A400 I/O data rates.

The results indicate no major discrepancies lie in the HOSC system model used for the algorithm development. The BASIC program has been emphasized since it is more transportable than SLAM or PASCAL. However, for a next generation simulation model, PASCAL will be contructed in a user friendly format since it has some features which make it suitable for this type of simulation.

3.4 Conclusions

Results of a fair run simulation using the configuration of Figure 9 has been tabulated in Table 7. The purpose of this comparison was to determine:

TABLE 7

COMPARISON OF FOUR SIMULATION RUNS
(BASIC PROGRAM)

P.D.S.	TTNR(MS)	LONGEST WAITING TIME (MS, % TTNR)					AVERAGE WAITING TIME (%)					BUS BUSY TIME (%)				
		RUN A	RUN B	RUN C	RUN D	RUN E	RUN A	RUN B	RUN C	RUN D	RUN E	RUN A	RUN B	RUN C	RUN D	RUN E
1 1 1	136	128 (94%)	128 (94%)	256 (188%)	128 (94%)	8 (6%)	29	27	45	29	39					
2 1 1	1000	128 (13%)	128 (13%)	118 (11.8%)	19 (2%)	4 (.4%)	2.8	4.1	4.8	1.7	.22	6.75%	6.34%	6.32%	6.35%	5.9%
2 2 1	1000	45 (5%)	45 (5%)	8.4 (.84%)	8 (.8%)	9 (1%)	1.5	1.7	.35	.47	.42					
3 1 1	512	129 (25%)	128 (25%)	256 (50%)	116 (23%)	5 (1%)	1.9	10.9	11.5	1.6	.21					
3 2 1	512	128 (25%)	130 (26%)	258 (50%)	116 (23%)	4 (1%)	3.2	1.2	.85	1.2	.57					
3 3 1	1000	101 (10%)	101 (10%)	202 (20%)	101 (10%)	13 (1.3%)	3.0	3.0	2.8	3.2	.81					

P.D.S = PORT, DEVICE, SOURCE

Average Waiting Time is expressed as percent of TTNR

TTNR = Time to Next Transmission Request = Time Between Data Transfer Requests

Number of Collisions: RUN A - 10; RUN B - 6; RUN C - 14; RUN D - 3; RUN E - 3

RUN A	1500 data transfers, Data file dump once every 12 seconds, 64,000 Bytes each dump
RUN B	1333 data transfers, Data file dump once every 12 seconds, 64,000 Bytes each dump
RUN C	1000 data transfers, Data file dump once every 12 seconds, 128,000 Bytes each dump
RUN D	500 data transfers, Data file dump once every 12 seconds, 64,000 Bytes each dump
RUN E	500 data transfers, Data file dump once every 120 seconds, 64,000 Bytes each dump (For a 500 iteration run time of 36 milliseconds no data file dump occurs).

ORIGINAL PAGE IS
OF POOR QUALITY

a) The number of iterations necessary to produce consistent results.

b) Determine the effects of varying the data file dump interval.

Since each simulation run uses random number generators as part of the program, it should not be surprising to see small differences in the output summary statistics. Indeed two runs of the same configuration with the same number of iterations will produce slightly different results. This is completely expected and does not reduce the value of the results at all.

The results tabulated in Table 7 indicate that the system performs as desired with one exception. The data file dump. Every 24 seconds, Run C, creates a waiting time of 256 milliseconds for P.D.S. 111, which is 188% of the time between transmission requests for source P.D.S. 111. The impact of these results from a realization that 100% waiting time for a source would mean that source has been waiting for an opportunity to transmit for such a long time that it now has three transmission requests rather than one. It is obvious from the data in Table 7 that large data file dumps will tend to create a log jam for devices with active external data sources. These external data sources may not be extinguishable; hence, the need to provide sufficient data storage in the device to hold incoming data during a large data file dump is paramount.

An analytical feeling for this problem is easily derived. Any source with a data file dump will experience an external source transmission request whenever the data file dump time exceeds the total time between transmission requests for that source. In the configuration of Figure 9, we have a data file dump from P.D. 11 to P.D. 41. The I/O rate for P.D. 11 is 500 K Bytes per second and the I/O rate for P.D. 41 is 3.30 M Bytes per second. The channel transfer rate is 6.25 M Bytes per second so that data file dump will take a total time equal to the channel setup release time plus the time to dump X Bytes at the slowest I/O rate or for our configurations the data file dump time (DFDT) is

$$\text{DFDT} = 25\ \mu\text{sec} + X(2\ \mu\text{sec/Byte}) \ .$$

For 64,000 Bytes DFDT = 128 msec and
for 128,000 Bytes DFDT = 256 msec.
In fact whenever the DFDT exceeds the fastest source average time to transmission request time a problem will definitely arise. For the example where the number of bytes in a data file dump exceeds

$$\begin{matrix}\text{Number of Bytes}\\ \text{In Data File Dump}\end{matrix} = \frac{126.533 - .025}{2 \times 10^{-3}} = 68254 \text{ Bytes,}$$

the fastest source will possibly incur two transmission requests.

There are several ways to correct this situation:

a) Increase the buffer size of the source so it can store more than two full sets of data between transmission opportunities. This is a viable option since source P.D.S. 111 is the most active with highest outside data arrival rate and only 2048 bytes to be accumulated between transmission requests.

b) Perform data file dumps more often but transfer proportionally less bytes per dump thereby reducing any sources waiting time. This may not be a viable option due to the lack of data file data being ready in a somewhat steady occurrance rate. Also this would require a data file storage medium at the destinations; however, this is usually the case.

c) Break up the data file dump by transmitting it in smaller packets. That is rather than 64,000 bytes in a steady stream for a short time once every 12 seconds, transmit 8,000 bytes, break and release channel to allow another user but request transmission rights immediately and repeat for 8 times. This would transfer the data in almost the same time as sending all 64,000 bytes while allowing the active devices a chance to clear their stored data.

All the above options could be accomplished through software programming of the source device P.D. 11; thus, it is a HOSC system operators choice of which method to utilize.

4.0 LITERATURE SURVEY

4.1 References

1. Wilkinson, Bill. Telephone Conversation, Aug. 12, 1982. Network System Corporation, Huntsville, AL (205) 882-2366.

2. Jobe, Sherman. Telephone Conversation, Aug. 30, 1982. MSFC, Huntsville, AL. (205) 453-5320.

3. "Shuttle Program, HOSC Operations Procedure Document," SPMD1.2.1. NASA Marshall Space Flight Center, 1979, p. 1.6.

4. VAX Hardware Handbook. Maynard Mass: Digital Equipment Corporation, 1980.

5. McMullen, Frank. Telephone Conversation, Sept. 16, 1982. Digital Equipment Corporation, New Orleans, LA, (504) 888-7230.

6. "Model 3240 Processors," Product Information Sheet. Oceanport, NJ: Perkin Elmer Corporation, 1981.

7. Staggs, Clint. Telephone Conversation, Sept. 17, 1982. Perkin Elmer Corporation, Huntsville, AL, (205) 533-6123.

8. A400 Adapter Reference Manual, 42990008. Brooklyn Park, MN: Network Systems Corp. 1979.

9. Performance of HYPER channel Networks: Parameter Measurements, Models and Analysis. Technical Report 82-3, WR Franta and John Harth. Univ. of Minn, 1982.

10. Nucleus Adapter Reference Manual, 4290003. Brooklyn Park, MN: Network Systems, Corp. 1980.

4.2 Bibliography

4.2.1 Publications

1. Abramson, Norman. The ALOHA System and the Alternative for Computer Communication". Proc. Fall Joint Computer Conference, AFIPS Conference Proceedings, Vol. 37, pp. 281-285, 1970.

2. Abramson, Norman. "The Throughput of Packet Broadcasting Channels," IEEE Transactions on Communications. Vol. COM-25. No. 1. January 1977.

3. Donnelly, James E., and Jeffery W. Yh. "Interaction Between Protocol Level in a Prioritized CSMA Broadcast Network," Computer Networks. North Holland Publishing Co., 1979. pp. 9-23.

4. Franta, W. R. and John Heath. "Performance of HYPER Channel Networks: Parameters, Measurements, models and Analysis," TR-82-3. Minneapolis, Minn: University ofMinnesota Computer Science Dept., 1982.

5. Franta, William and Mark Bennedict Bilodeau. "Analysis of a Prioritized CSMA Protocol Based on Staggered Delays," Acta Informatica, Vol. 13, 1980. pp. 299-324.

6. Hall, Ronald C., and Denish H. Widman. "Implementation of a Distributed Computing Gateway (DCG) at Sandia National Laboratories," SAND 82-0490. Albuquerque, NM: Sandia National Laboratories, 1982.

7. Kleinrock, Leonard and Simon Lam. "Packet Switching in a Multi access Broadcast Channel: Performance Evaluation," IEEE Transactions on Communications, Vol. Comm-23, No. 4, April 1975.

8. Metclafe, R. M. and D. R. Boggs. "Ethernet: Distributed Packet Switching for Local Computer Networks," CACM 19, No. 7, July 1976, pp. 395-404.

4.2.2 Reference Manuals (by Manufacturer)

1. VAX Architecture. Maynard, MA: Digital Equipment Corp. 1981.

2. VAX Hdwr Handbook. Maynard, MA: Digital Equipment Corporation, 1980.

3. VAX Software Handbook. Maynard, MA: Digital Equipment Corp. 1981-82.

4. "Shuttle Program, HOSC Operations Procedures Document," SPMD 1.2.1 Rev. 6, Huntsville, AL: NASA Marshall Space Flight Center, 1982.

5. A400 Adapter Reference Manual, 42990008. Brooklyn Park, MN: Network Systems Corporation, 1979.

6. "HYPER Channel Systems Description," A01-000002 Brooklyn Park, MN: Network Systems Corporation, 12/1980.

7. Nucleus Adapter Reference Manual, 4290003. Brooklyn Park, MN: Network Systems Corporation, 1980.

8. PI 13/14 Peripheral Interface Reference Manual, 42990015. Brooklyn Park, MN: Network Systems Corp., 1981.

9. "Model 3240 Processors," Product Information Sheet. Ocean Port, NJ: Perkin Elmer Corporation, 1981.

10. 3240 Processor Maintenance Manual, H29-683. Ocean Port, NJ: Perkin Elmer, 1979.

APPENDIX I

BASIC SIMULATION ALGORITHM PROGRAM

A listing of the BASIC simulation algorithm program is presented in this appendix. The program contains plentiful comments and is user oriented, with prompts and options displayed on the interactive screen. The BASIC language is common to many machines; however, the input output commands are usually particular to a single machine, in this case the HP-87 system.

```
0 ! ************************************************
10 ! *  PROGRAM:           HOSC                       *
0 ! *  AUTHOR:            DR. FRANK INGELS           *
0 ! *  PROGRAMMER:        TERESA BENNET              *
0 ! *  DATE:              MAY 12,1983                *
0 ! ************************************************
0 ! I   --  # OF PORTS                       I<=3
0 ! IJ --  # OF DEVICES FOR EACH PORT        J<=5
0 ! IJK--  # OF SOURCES FOR EACH DEVICE      K<=9
00 ! LAI(J,K)  --  AVERAGE ARRIVAL RATE OF DATA TO SOURCE
10 ! AQI(J.K)    --  # OF BYTES ACCUMULATED BY SOURCE K, DEVICE J, PORT I
20 !                  BEFORE DEVICE J REQUESTS A TRANSMISSION
30 ! TSR  --  CHANNEL SETUP AND RELEASE TIME
40 ! CR   --  CHANNEL DATA TRANSFER RATE
50 ! PR(IJK,LMN) -- PROBABILTIY THAT SOURCE IJK WILL TRANSMIT DATA TO LMN
60 ! BBT = BUS BUSY TIME
70 ! BEGIN PROGRAM EXECUTION
80 INTEGER I,J,K.L.M.N.O
90 BBT=0 @ CH1=999999 @ CH2=999999 @ CLSN$="N" @ CLN$="N" @ EG$="N" @ AF$="F"
00 DIM PDS(200),PRO(200),T(200),TIMESTT(200),DT(200),BYTES(20),TIM(20),DTFL(20)
WT(20)
10 CLEAR @ DISP TAB (20);"MENU" @ DISP @ DISP "1  --  RECALL AN EXISTING DATA F
LE"
20 DISP "2  --  EXECUTE PROGRAM WITHOUT THE FILE OPTION"
30 DISP "3  --  CREATE A NEW DATA FILE AND EXECUTE PROGRAM"
40 DISP @ DISP "ENTER YOUR SELECTION ";@ INPUT AN
50 IF AN<1 OR AN>3 THEN 210
60 PRINTER IS 701
70 DISP " " @ DISP "INDICATE THE NUMBER OF PROGRAM ITERATIONS ";@ INPUT NUMITER

80 ON AN GOTO 290,310,290
90 DISP @ DISP "ENTER THE DATA FILE NAME ";@ INPUT DFNAME$
00 IF AN=1 THEN GETDATA=999 @ GOTO 1350 ELSE GETDATA=888
05 ! ***********************************************************
06 ! DATA ENTRY ROUTINE  (IF AN=2 OR AN=3)
07 ! LINES 310-1330
08 ! ***********************************************************
10 CLEAR @ DISP "PLEASE ENTER THE FOLLOWING DATA ITEMS:" @ DISP
20 DISP "# OF HYPERCHANNEL PORTS       ";@ INPUT HY
30 FOR I=1 TO HY @ DISP "# OF DEVICES FOR PORT ";I;" ";@ INPUT DV(I)@ NEXT I @
1 @ CLEAR
40 FOR K=1 TO DV(I)
350 DISP "# OF SOURCES FOR PORT ";I;" DEVICE ";K;@ INPUT SR(I,K)
360 NEXT K @ I=I+1 @ IF I<= HY THEN 340 ELSE I=0
```

```
370 I=I+1 @ CLEAR @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ CLEAR
380 DISP " PORT ";I;" DEVICE ";J;" SOURCE ";K @ DISP @ DISP
 30 DISP "AVERAGE ARRIVAL RATE "; @ INPUT LA(J,K)
 00 DISP "# OF BYTES ACCUMULATED BEFORE A TRANSMISSION REQUEST IS TO BE MADE BY
EVICE ";J; @ DISP "(VALUE ASSIGNED 500 BYTES UNLESS";
 10 Q$="1" @ QQ(J,K)=500 @ DISP " OTHERWISE SECIFIED)"; @ INPUT Q$
 20 IF Q$ <> "" THEN QQ(J,K)=VAL (Q$)
 30 NEXT K @ NEXT J @ ON I GOSUB 450,490,530,570,610,650,690,730,770
 40 IF I=HY THEN 810 ELSE GOTO 370
 50 FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ LA1(J,K)=LA(J,K) @ LA(J,K)=0 @ Q1(J,
 )=QQ(J,K) @ QQ(J,K)=0
 60 IF LA1(J,K)=0 THEN ATR1(J,K)=999999 ELSE ATR1(J,K)=Q1(J,K)/LA1(J,K)
 70 NEXT K @ NEXT J @ RETURN
 80 LA=LA1(J,K) @ QQ=Q1(J,K) @ ATR=ATR1(J,K) @ RETURN
 90 FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ LA2(J,K)=LA(J,K) @ LA(J,K)=0 @ Q2(J,
 )=QQ(J,K) @ QQ(J,K)=0
 00 IF LA2(J,K)=0 THEN ATR2(J,K)=999999 ELSE ATR2(J,K)=Q2(J,K)/LA2(J,K)
 10 NEXT K @ NEXT J @ RETURN
 20 LA=LA2(J,K) @ QQ=Q2(J,K) @ ATR=ATR2(J,K) @ RETURN
 30 FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ LA3(J,K)=LA(J,K) @ LA(J,K)=0 @ Q3(J,
 )=QQ(J,K) @ QQ(J,K)=0
 40 IF LA3(J,K)=0 THEN ATR3(J,K)=999999 ELSE ATR3(J,K)=Q3(J,K)/LA3(J,K)
 50 NEXT K @ NEXT J @ RETURN
 60 LA=LA3(J,K) @ QQ=Q3(J,K) @ ATR=ATR3(J,K) @ RETURN
 70 FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ LA4(J,K)=LA(J,K) @ LA(J,K)=0 @ Q4(J,
 )=QQ(J,K) @ QQ(J,K)=0
 80 IF LA4(J,K)=0 THEN ATR4(J,K)=999999 ELSE ATR4(J,K)=Q4(J,K)/LA4(J,K)
 90 NEXT K @ NEXT J @ RETURN
 00 LA=LA4(J,K) @ QQ=Q4(J,K) @ ATR=ATR4(J,K) @ RETURN
 10 FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ LA5(J,K)=LA(J,K) @ LA(J,K)=0 @ Q5(J,
 )=QQ(J,K) @ QQ(J,K)=0
 20 IF LA5(J,K)=0 THEN ATR5(J,K)=999999 ELSE ATR5(J,K)=Q5(J,K)/LA5(J,K)
 30 NEXT K @ NEXT J @ RETURN
 40 LA=LA5(J,K) @ QQ=Q5(J,K) @ ATR=ATR5(J,K) @ RETURN
 50 FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ LA6(J,K)=LA(J,K) @ LA(J,K)=0 @ Q6(J,
 )=QQ(J,K) @ QQ(J,K)=0
 60 IF LA6(J,K)=0 THEN ATR6(J,K)=999999 ELSE ATR6(J,K)=Q6(J,K)/LA6(J,K)
 70 NEXT K @ NEXT J @ RETURN
 80 LA=LA6(J,K) @ QQ=Q6(J,K) @ ATR=ATR6(J,K) @ RETURN
 90 FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ LA7(J,K)=LA(J,K) @ LA(J,K)=0 @ Q7(J,
 )=QQ(J,K) @ QQ(J,K)=0 @ NEXT K @ NEXT J @ RETURN
 00 IF LA7(J,K)=0 THEN ATR7(J,K)=999999 ELSE ATR7(J,K)=Q7(J,K)/LA7(J,K)
 10 NEXT K @ NEXT J @ RETURN
 20 LA=LA7(J,K) @ QQ=Q7(J,K) @ ATR=ATR7(J,K) @ RETURN
730 FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ LA8(J,K)=LA(J,K) @ LA(J,K)=0 @ Q8(J,
K)=QQ(J,K) @ QQ(J,K)=0 @ NEXT K @ NEXT J @ RETURN
```

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

```
740 IF LA8(J,K)=0 THEN ATR8(J,K)=999999 ELSE ATR8(J,K)=Q8(J,K)/LA8(J,K)
750 NEXT K @ NEXT J @ RETURN
760 LA=LA8(J,K) @ QQ=Q8(J,K) @ ATR=ATR8(J,K) @ RETURN
 70 FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ LA9(J,K)=LA(J,K) @ LA(J,K)=0 @ Q9(J.
 )=QQ(J,K) @ QQ(J,K)=0 @ NEXT K @ NEXT J @ RETURN
 80 IF LA9(J,K)=0 THEN ATR9(J,K)=999999 ELSE ATR9(J,K)=Q9(J,K)/LA9(J,K)
 90 NEXT K @ NEXT J @ RETURN
 00 LA=LA9(J,K) @ QQ=Q9(J,K) @ ATR=ATR9(J,K) @ RETURN
 10 CLEAR @ FOR I=1 TO HY @ FOR J=1 TO DV(I)
 20 DISP "PROPAGATION DISTANCE BETWEEN PORT ";I;" AND DEVICE ";J;" IN MICROSECON
 S";@ INPUT D(I,J)
 30 NEXT J @ NEXT I
 40 CLEAR @ FOR I=1 TO HY @ DISP "WAIT TIME OF PORT ";I;" BEFORE TRANSMISSION RE
 UEST (IN EVENT OF A COLLISION)";@ INPUT W(I)
 50 NEXT I
 60 CLEAR @ DISP "CHANNEL SETUP AND RELEASE TIME ASSIGNED THE VALUE 25 MICROSECO
 DS"
 70 Q$=".000025" @ DISP " " @ DISP "PRESS <END LINE> OR ENTER THE PROPER VALUE "
 @ INPUT Q$@ IF Q$ <> "" THEN TSR=VAL (Q$) ELSE TSR=.000025
 80 DISP " " @ DISP "CHANNEL DATA TRANSFER RATE IS 6.25MBS" @ DISP " "
 90 Q$="6250000" @ DISP "PRESS <END LINE> OR ENER THE PROPER VALUE ";@ INPUT Q$@
 IF Q$ <> "" THEN CR=VAL (Q$) ELSE CR=6250000
 00 CLEAR @ FOR I=1 TO HY @ FOR J=1 TO DV(I)
 10 DISP "RATE AT WHICH DATA IS TRANSFERED FROM DEVICE ";J;" TO PORT ";I;
 20 INPUT R(I,J)
 30 LT(I,J)=1/R(I,J)
 40 NEXT J @ NEXT I
 50 GOSUB 960 @ GOTO 1080
 60 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
 70 ON I GOSUB 990,1000,1010,1020,1030,1040,1050,1060,1070
 80 NEXT K @ NEXT J @ NEXT I @ RETURN
 90 MNEG1(J,K)=0 @ TNG1(J,K)=0 @ TN1(J,K)=0 @ COL1(J,K)=0 @ RETURN
)00 MNEG2(J,K)=0 @ TNG2(J,K)=0 @ TN2(J,K)=0 @ COL2(J,K)=0 @ RETURN
)10 MNEG3(J,K)=0 @ TNG3(J,K)=0 @ TN3(J,K)=0 @ COL3(J,K)=0 @ RETURN
)20 MNEG4(J,K)=0 @ TNG4(J,K)=0 @ TN4(J,K)=0 @ COL4(J,K)=0 @ RETURN
)30 MNEG5(J,K)=0 @ TNG5(J,K)=0 @ TN5(J,K)=0 @ COL5(J,K)=0 @ RETURN
)40 MNEG6(J,K)=0 @ TNG6(J,K)=0 @ TN6(J,K)=0 @ COL6(J,K)=0 @ RETURN
)50 MNEG7(J,K)=0 @ TNG7(J,K)=0 @ TN7(J,K)=0 @ COL7(J,K)=0 @ RETURN
)60 MNEG8(J,K)=0 @ TNG8(J,K)=0 @ TN8(J,K)=0 @ COL8(J,K)=0 @ RETURN
)70 MNEG9(J,K)=0 @ TNG9(J,K)=0 @ TN9(J,K)=0 @ COL9(J,K)=0 @ RETURN
1080 NUMPROB=0
1090 PDS(0)=0 @ PRO(0)=0 @ FOR I=1 TO 200 @ CLEAR @ DISP "PREVIOUS COMBINATION "
;PDS(I-1);"   PROBABILITY ";PRO(I-1) @ DISP " "
1100 DISP "SELECT (PORT,DEVICE,SOURCE) WHICH TRANSMITS TO (PORT,DEVICE,SOURCE) "
```

```
1110 DISP "FORMAT <PDSPDS>. THESE COMBINATIONS MUST BE ENTERED IN ORDER. "
1120 DISP "ENTER <0> IF FINISHED ENTERING PROBABILITIES ";@ INPUT PDS(I)@ TIMEST
  I)=0
  30 IF PDS(I)=0 THEN 1260 ELSE NUMPROB=NUMPROB+1
  40 DISP " " @ DISP "PROBABILITY OF OCCURRENCE ";@ INPUT PRO(I)
  50 GOSUB 1160 @ GOTO 1250
  60 J=INT (PDS(I)/100000) @ K=INT (PDS(I)/10000)-J*10
  70 L=-(J*100)-K*10+INT (PDS(I)/1000) @ M=-(J*1000)-K*100-L*10+INT (PDS(I)/100)

  80 N=-(J*10000)-K*1000-L*100-M*10+INT (PDS(I)/10)
  90 O=-(J*100000)-K*10000-L*1000-M*100-N*10+PDS(I)
  00 A=J @ J=K @ K=L @ ON A GOSUB 480,520,560,600,640,680,720,760,800
  10 CP1=QQ*LT(A,J)
  20 CP2=QQ*LT(M,N)
  30 IF CP1>CP2 THEN TL=CP1 ELSE TL=CP2
  40 T(I)=TSR+2000/CR+TL @ RETURN
  50 NEXT I @ PREJ=99 @ PREK=99
  60 SNUM=0 @ PREJ=99 @ PREK=99 @ CLEAR @ FOR I=1 TO 200 @ CLEAR @ DISP "ENTER I
 IJ-LM COMBINATIONS WHICH HAVE SOURCE DATA FILE DUMPS."
  70 DISP "ENTER A <0> WHEN COMPLETED." @ DISP " " @ DISP "SOURCE TO DESTINATION
      ";@ INPUT DTFL(I)@ IF DTFL(I)=0 THEN 1330
  80 WT(I)=0 @ SNUM=SNUM+1 @ J=INT (DTFL(I)/1000) @ K=DTFL(I)-J*1000 @ K=INT (K/
  0)
  90 IF J=PREJ AND K=PREK THEN BYTES(I)=BYTES(I-1) @ TIM(I)=TIM(I-1) @ GOTO 1320

  00 PREJ=J @ PREK=K @ DISP "# OF BYTES TO BE DUMPED      ";@ INPUT BYTES(I)
  10 DISP "TIME INTERVAL BETWEEN DUMPS (IN SECONDS)";@ INPUT TIM(I)
  20 NEXT I
  30 IF AN=2 THEN 1720
  34 ! *****************************************************************
  35 ! READ OR WRITE DATA TO THE DATA FILE (GETDATA = 999 -READ OR 888 -WRITE)
  36 ! LINES 1340-1710
  37 ! *****************************************************************
  40 CLEAR @ CREATE DFNAME$,2123.8
  50 DFNM$=DFNAME$&":D700" @ ASSIGN# 1 TO  DFNM$
  60 IF GETDATA=999 THEN READ# 1,1 : HY ELSE PRINT# 1,1 : HY
  70 II=1 @ FOR J=1 TO HY @ II=II+1
  80 IF GETDATA=999 THEN READ# 1,II : DV(J) ELSE PRINT# 1,II : DV(J)
  90 NEXT J @ FOR I=1 TO HY @ FOR J=1 TO DV(I)
  00 II=II+1 @ IF GETDATA=999 THEN READ# 1,II : SR(I,J) ELSE PRINT# 1,II : SR(I,

1410 NEXT J @ NEXT I
1420 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
1430 IF GETDATA=888 THEN ON I GOSUB 480,520,560,600,640,680,720,760,800
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
1440 II=II+1
1450 IF GETDATA=999 THEN READ# 1.II : QQ(J.K) ELSE PRINT# 1.II : QQ
1460 II=II+1
1470 IF GETDATA=999 THEN READ# 1.II : LA(J.K) ELSE PRINT# 1.II : LA
1480 NEXT K @ NEXT J @ IF GETDATA=999 THEN ON I GOSUB 450.490.530.570.610.650.69
 .730.770
 490 NEXT I
 500 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ II=II+1
 510 IF GETDATA=999 THEN READ# 1.II ; D(I.J) ELSE PRINT# 1.II ; D(I.J)
 520 NEXT J @ NEXT I
 530 FOR I=1 TO HY @ II=II+1 @ IF GETDATA=999 THEN READ# 1.II ; W(I) ELSE PRINT#
 1.II ; W(I)
 540 NEXT I
 550 II=II+1 @ IF GETDATA=999 THEN READ# 1.II ; TSR ELSE PRINT# 1.II ; TSR
 560 II=II+1 @ IF GETDATA=999 THEN READ# 1.II ; CR ELSE PRINT# 1.II ; CR
 570 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ II=II+1
 580 IF GETDATA=999 THEN READ# 1.II ; R(I.J)@ LT(I.J)=1/R(I.J) ELSE PRINT# 1.II
 R(I.J)
 590 NEXT J @ NEXT I
 600 II=II+1 @ IF GETDATA=999 THEN READ# 1.II ; NUMPROB ELSE PRINT# 1.II ; NUMPR
 B
 610 FOR I=1 TO NUMPROB @ II=II+1 @ IF GETDATA=999 THEN READ# 1.II ; PDS(I) ELSE
 PRINT# 1.II ; PDS(I)
 620 II=II+1 @ IF GETDATA=999 THEN READ# 1.II ; PRO(I) ELSE PRINT# 1.II ; PRO(I)

 630 IF GETDATA=999 THEN TIMESTT(I)=0 @ GOSUB 1160
 640 NEXT I
 650 II=II+1 @ IF GETDATA=999 THEN READ# 1.II ; SNUM ELSE PRINT# 1.II ; SNUM
 660 FOR I=1 TO SNUM @ WT(I)=0 @ II=II+1 @ IF GETDATA=999 THEN READ# 1.II ; DTFL(
 I) ELSE PRINT# 1.II ; DTFL(I)
 670 II=II+1 @ IF GETDATA=999 THEN READ# 1.II ; BYTES(I) ELSE PRINT# 1.II ; BYTE
 (I)
 680 II=II+1 @ IF GETDATA=999 THEN READ# 1.II ; TIM(I) ELSE PRINT# 1.II ; TIM(I)

 690 NEXT I
 700 ASSIGN# 1 TO *
 710 GOSUB 960
 720 CLEAR @ DISP TAB (20);"MENU" @ DISP " " @ DISP "1  --  PRINT OUT CURRENT DA
 A " @ DISP "2  --  CORRECT DATA ERRORS "
 730 DISP "3  --  RUN PROGRAM " @ DISP " " @ DISP "ENTER YOUR SELECTION ";@ INPU
 SELECT
1740 IF SELECT<1 OR SELECT>3 THEN 1720
1750 IF SELECT=3 THEN 1760 ELSE ON SELECT GOTO 3840.4210
1760 SAVEATR=99999 @ SAVELA=0
1770 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I.J)
1780 ON I GOSUB 480.520.560.600.640.680.720.760.800
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
1790 IF LA>SAVELA THEN SAVELA=LA @ SAVEIJK=I*100+J*10+K
1800 IF ATR=0 THEN 1820
1810 IF ATR<SAVEATR THEN SAVEATR=ATR @ SAVEATRIJK=I*100+J*10+K
1820 IF ATR=999999 THEN TLST=999999 ELSE TLST=0
1830 ON I GOSUB 1850,1860,1870,1880,1890,1900,1910,1920,1930
 840 NEXT K @ NEXT J @ NEXT I @ GOTO 1940
 850 TLST1(J,K)=TLST @ RETURN
 860 TLST2(J,K)=TLST @ RETURN
 870 TLST3(J,K)=TLST @ RETURN
 880 TLST4(J,K)=TLST @ RETURN
 390 TLST5(J,K)=TLST @ RETURN
 300 TLST6(J,K)=TLST @ RETURN
 310 TLST7(J,K)=TLST @ RETURN
 320 TLST8(J,K)=TLST @ RETURN
 330 TLST9(J,K)=TLST @ RETURN
 340 PRINT " " @ PRINT "FIRST USER IS ";SAVEIJK @ ITERNUM=0 @ NUMDEST=0
 350 PRINT " " @ PRINT "TOTAL TIME GONE BY ";SAVEATR @ CN=0 @ TTGB=SAVEATR
 360 CLEAR @ DISP "                PRINTER MENU" @ DISP " " @ DISP "1  --  NO PRIN
 JUT GIVEN ON EACH CYCLE"
 370 DISP "2  --  PRINTOUT GIVEN ON EACH CYCLE" @ DISP "3  --  SPECIFY THE LAST
 CYCLES TO BE PRINTED" @ DISP " "
 380 INPUT PR@ IF PR<1 OR PR>3 THEN CLEAR @ GOTO 1960
 390 IF PR=2 THEN PRINTER IS 701 ELSE PRINTER IS 1
 )00 IF PR=3 THEN DISP "ENTER THE LAST X CYCLES WHICH ARE TO BE PRINTED ";@ INPU
 NUMCYCLES
 )05 ! *****************************************************************
 )06 ! GENERATE A RANDOM NUMBER, THEN USE IT TO SELECT THE DESINATION DEVICE
 )07 ! LINES 2010-2140
 )08 ! *****************************************************************
 )10 XXX=RND
 )20 IF PR=3 THEN 2030 ELSE 2040
 )30 IF ITERNUM>= NUMITER-NUMCYCLES THEN PRINTER IS 701
 )40 PRINT " " @ PRINT " " @ PRINT "ITERATION NUMBER ";ITERNUM+1 @ PRINT " "
 )50 FOR I=1 TO NUMPROB
 )60 IJK=INT (PDS(I)/1000)
 )70 IF IJK=SAVEIJK THEN NUMDEST=NUMDEST+1 @ SAVEEND=I ELSE 2090
 )80 IF NUMDEST=1 THEN START=I @ PROB=PRO(I) @ PRINT "PROBABILITY ";PROB
 )90 NEXT I @ PRINT "RANDOM NUMBER ";XXX
 100 IF XXX<PROB THEN SOURCE=START @ GOTO 2140
 110 FOR I=START TO SAVEEND
 120 IF XXX<= PROB THEN SOURCE=I @ GOTO 2140
2130 PROB=PROB+PRO(I+1) @ PRINT "PROBABILITY ";PROB @ NEXT I
2140 PRINT " " @ PRINT "DESTINATION ";PDS(SOURCE)-SAVEIJK*1000
2150 PRINT " " @ PRINT "PORT,DEVICE,SOURCE         TOTAL TIME GONE BY SINCE LAST
 TRANSMISSION" @ TTGBSLT=SAVEATR
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
2160 TIMESTT(SOURCE)=TIMESTT(SOURCE)+1
2170 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
2180 TLST=999999 @ ON I GOSUB 2210,2220,2230,2240,2250,2260,2270,2280,2290
2190 PRINT I;"  ";J;"  ";K;TAB (30);TLST
2200 NEXT K @ NEXT J @ NEXT I @ GOTO 2300
2210 IF TLST1(J,K)=999999 THEN RETURN ELSE TLST1(J,K)=TLST1(J,K)+TTGBSLT @ TLST=
TLST1(J,K) @ RETURN
2220 IF TLST2(J,K)=999999 THEN RETURN ELSE TLST2(J,K)=TLST2(J,K)+TTGBSLT @ TLST=
TLST2(J,K) @ RETURN
2230 IF TLST3(J,K)=999999 THEN RETURN ELSE TLST3(J,K)=TLST3(J,K)+TTGBSLT @ TLST=
TLST3(J,K) @ RETURN
2240 IF TLST4(J,K)=999999 THEN RETURN ELSE TLST4(J,K)=TLST4(J,K)+TTGBSLT @ TLST=
TLST4(J,K) @ RETURN
2250 IF TLST5(J,K)=999999 THEN RETURN ELSE TLST5(J,K)=TLST5(J,K)+TTGBSLT @ TLST=
TLST5(J,K) @ RETURN
2260 IF TLST6(J,K)=999999 THEN RETURN ELSE TLST6(J,K)=TLST6(J,K)+TTGBSLT @ TLST=
TLST6(J,K) @ RETURN
2270 IF TLST7(J,K)=999999 THEN RETURN ELSE TLST7(J,K)=TLST7(J,K)+TTGBSLT @ TLST=
TLST7(J,K) @ RETURN
2280 IF TLST8(J,K)=999999 THEN RETURN ELSE TLST8(J,K)=TLST8(J,K)+TTGBSLT @ TLST=
TLST8(J,K) @ RETURN
2290 IF TLST9(J,K)=999999 THEN RETURN ELSE TLST9(J,K)=TLST9(J,K)+TTGBSLT @ TLST=
TLST9(J,K) @ RETURN
2300 GOSUB 2310 @ GOTO 3020
2305 ! ************************************************************************
2306 ! FIND SMALL & SMALL2 TO BE USED TO CHECK FOR COLLISIONS AND NRTT
2307 ! LINES 2310-2540
2308 ! ************************************************************************
2310 PRINT " " @ PRINT "PORT,DEVICE,SOURCE        REMAINING TIME TO TRANSMIT (NO
RMALIZED FASHION)"
2320 SMALL=999988 @ SAVESM=999988 @ SMALL2=99988 @ SAVESM2=99988
2330 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
2340 ON I GOSUB 480,520,560,600,640,680,720,760,800
2350 ON I GOSUB 2370,2380,2390,2400,2410,2420,2430,2440,2450
2360 GOTO 2460
2370 TLST=TLST1(J,K) @ RETURN
2380 TLST=TLST2(J,K) @ RETURN
2390 TLST=TLST3(J,K) @ RETURN
2400 TLST=TLST4(J,K) @ RETURN
2410 TLST=TLST5(J,K) @ RETURN
2420 TLST=TLST6(J,K) @ RETURN
2430 TLST=TLST7(J,K) @ RETURN
2440 TLST=TLST8(J,K) @ RETURN
2450 TLST=TLST9(J,K) @ RETURN
```

```
2460 IF ATR=999999 THEN NRTT=999999 ELSE NRTT=ATR-TLST
2470 IF CLSN$="Y" THEN SUB=-1 ELSE SUB=1
2480 ON I GOSUB 2550,2600,2650,2700,2750,2800,2850,2900,2950
2490 IF CLSN$="Y" THEN 2510
2500 PRINT I;"  ";J;"  ";K;TAB (30);NRTT
 510 IF NRTT<SMALL THEN CH1=SMALL @ SMALL=NRTT @ CH2=SAVESM @ SAVESM=I*100+J*10+
  @ EQ$="Y"
 520 IF CH1<SMALL2 THEN SMALL2=CH1 @ SAVESM2=CH2
 530 IF NRTT=SMALL AND NRTT<SMALL2 AND EQ$="N" THEN SMALL2=NRTT @ SAVESM2=I*100+
 *10+K
 540 EQ$="N" @ NEXT K @ NEXT J @ NEXT I @ EQ$="N" @ RETURN
 550 NRTT1(J,K)=NRTT @ IF NRTT<0 THEN TNG1(J,K)=TNG1(J,K)+NRTT @ TN1(J,K)=TN1(J,
 )+SUB
 560 IF CLSN$ <> "Y" THEN 2580
 570 IF NRTT<0 THEN TNG1(J,K)=TNG1(J,K)-2*NRTT
 580 IF NRTT<MNEG1(J,K) THEN MNEG1(J,K)=NRTT
 590 RETURN
 600 NRTT2(J,K)=NRTT @ IF NRTT<0 THEN TNG2(J,K)=TNG2(J,K)+NRTT @ TN2(J,K)=TN2(J,
 )+SUB
 610 IF NRTT<MNEG2(J,K) THEN MNEG2(J,K)=NRTT
 620 IF CLSN$ <> "Y" THEN 2640
 630 IF NRTT<0 THEN TNG2(J,K)=TNG2(J,K)-2*NRTT
 640 RETURN
 650 NRTT3(J,K)=NRTT @ IF NRTT<0 THEN TNG3(J,K)=TNG3(J,K)+NRTT @ TN3(J,K)=TN3(J,
 )+SUB
 660 IF CLSN$ <> "Y" THEN 2680
 670 IF NRTT<0 THEN TNG3(J,K)=TNG3(J,K)-2*NRTT
 680 IF NRTT<MNEG3(J,K) THEN MNEG3(J,K)=NRTT
 690 RETURN
 700 NRTT4(J,K)=NRTT @ IF NRTT<0 THEN TNG4(J,K)=TNG4(J,K)+NRTT @ TN4(J,K)=TN4(J,
 )+SUB
 710 IF CLSN$ <> "Y" THEN 2730
 720 IF NRTT<0 THEN TNG4(J,K)=TNG4(J,K)-2*NRTT
 730 IF NRTT<MNEG4(J,K) THEN MNEG4(J,K)=NRTT
 740 RETURN
 750 NRTT5(J,K)=NRTT @ IF NRTT<0 THEN TNG5(J,K)=TNG5(J,K)+NRTT @ TN5(J,K)=TN5(J,
 )+SUB
 760 IF CLSN$ <> "Y" THEN 2780
 770 IF NRTT<0 THEN TNG5(J,K)=TNG5(J,K)-2*NRTT
2780 IF NRTT<MNEG5(J,K) THEN MNEG5(J,K)=NRTT
2790 RETURN
2800 NRTT6(J,K)=NRTT @ IF NRTT<0 THEN TNG6(J,K)=TNG6(J,K)+NRTT @ TN6(J,K)=TN6(J,
K)+SUB
2810 IF CLSN$ <> "Y" THEN 2830
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
2820 IF NRTT<0 THEN TNG6(J.K)=TNG6(J.K)-2*NRTT
2830 IF NRTT<MNEG6(J.K) THEN MNEG6(J.K)=NRTT
2840 RETURN
2850 NRTT7(J,K)=NRTT @ IF NRTT<0 THEN TNG7(J,K)=TNG7(J,K)+NRTT @ TN7(J,K)=TN7(J.
 )+SUB
 360 IF NRTT<MNEG7(J.K) THEN MNEG7(J.K)=NRTT
 370 IF CLSN$ <> "Y" THEN 2890
 380 IF NRTT<0 THEN TNG7(J.K)=TNG7(J.K)-2*NRTT
 390 RETURN
 300 NRTT8(J.K)=NRTT @ IF NRTT<0 THEN TNG8(J.K)=TNG8(J.K)+NRTT @ TN8(J.K)=TN8(J.
 )+SUB
 310 IF NRTT<MNEG8(J.K) THEN MNEG8(J.K)=NRTT
 320 IF CLSN$ <> "Y" THEN 2940
 330 IF NRTT<0 THEN TNG8(J.K)=TNG8(J.K)-2*NRTT
 340 RETURN
 350 NRTT9(J.K)=NRTT @ IF NRTT<0 THEN TNG9(J.K)=TNG9(J,K)+NRTT @ TN9(J.K)=TN9(J.
 )+SUB
 360 IF NRTT<MNEG9(J.K) THEN MNEG9(J.K)=NRTT
 370 IF CLSN$ <> "Y" THEN 2990
 380 IF NRTT<0 THEN TNG9(J.K)=TNG9(J.K)-2*NRTT
 390 RETURN
 )05 ! ********************************************************************
 )06 ! THE COLLISION ALGORITHM.
 )07 ! LINES 3020 - 3300
 )08 ! ********************************************************************
 )20 AF$="F" @ IF INT ((PDS(SOURCE)-SAVESM*1000)/100)=INT (SAVESM/100) THEN AF$=
 I" @ PRINTER IS 701
 )21 IF AF$="I" THEN PRINT "TRANSMISSION ON SAME A400 SOURCE/DEST = ";PDS(SOURCE
  @ CLN$="N" @ GOTO 3260
 )22 IF CLN$="Y" THEN 3090
 )25 IF ABS (SMALL-SMALL2)<= .001 THEN PRINTER IS 701 ELSE GOTO 3260
 )40 PRINT "A COLLISION OCCURRED BETWEEN ";SAVESM;" AND ";SAVESM2
 )50 I=INT (SAVESM/100) @ L=INT (SAVESM2/100)
 )60 IF W(I)<= W(L) THEN 3090
 )70 SAVEIJK=SAVESM2 @ NUMDEST=0 @ SAVEATR=W(L) @ TIBG=TTGB+W(L) @ TIMESTT(SOUR(
 )=TIMESTT(SOURCE)-1 @ CLN$="Y"
 )80 PROB=0 @ CLSN$="Y" @ GOSUB 2320 @ CLSN$="N" @ GOTO 2010
 )90 CN=CN+1 @ PRINT "COLLISION NUMBER ";CN
 100 I=INT (SAVESM/100) @ J=INT (SAVESM/10)-I*10 @ K=SAVESM-I*100-J*10
3110 ON I GOSUB 3150,3160,3170,3180,3190,3200,3210,3220,3230
3120 I=INT (SAVESM2/100) @ J=INT (SAVESM2/10)-I*10 @ K=SAVESM2-I*100-J*10
3130 ON I GOSUB 3150,3160,3170,3180,3190,3200,3210,3220,3230
3140 GOTO 3240
```

```
3150 COL1(J,K)=COL1(J,K)+1 @ RETURN
3160 COL2(J,K)=COL2(J,K)+1 @ RETURN
 170 COL3(J,K)=COL3(J,K)+1 @ RETURN
 180 COL4(J,K)=COL4(J,K)+1 @ RETURN
 190 COL5(J,K)=COL5(J,K)+1 @ RETURN
 200 COL6(J,K)=COL6(J,K)+1 @ RETURN
 210 COL7(J,K)=COL7(J,K)+1 @ RETURN
 220 COL8(J,K)=COL8(J,K)+1 @ RETURN
 230 COL9(J,K)=COL9(J,K)+1 @ RETURN
 240 IF CLN$="Y" THEN DT(SOURCE)=T(SOURCE) @ CLN$="N" ELSE DT(SOURCE)=T(SOURCE)+
 (I) @ CLN$="N"
 250 GOTO 3300
 260 DT(SOURCE)=T(SOURCE)
 270 PRINT "NO COLLISION OCCURED " @ IF PR=1 THEN PRINTER IS 1
 280 IF PR=3 THEN 3290 ELSE 3300
 290 IF ITERNUM>= NUMITER-NUMCYCLES THEN 3300 ELSE PRINTER IS 1
 300 PRINT "DATA TRANSFER ";DT(SOURCE) @ IF AF$="T" THEN AF$="F" @ GOTO 3320
 310 BBT=BBT+DT(SOURCE)
 320 PRINT " " @ PRINT "PORT,DEVICE,SOURCE       ACCUMULATED WAITING TIMES SINCE
 THE LAST TRANSMISSION"
 330 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
 340 IF I*100+J*10+K=SAVEIJK THEN TLST=0 ELSE GOTO 3370
 350 ON I GOSUB 1850,1860,1870,1880,1890,1900,1910,1920,1930
 360 GOTO 3390
 370 TTGBSLT=DT(SOURCE) @ TLST=999999
 380 ON I GOSUB 2210,2220,2230,2240,2250,2260,2270,2280,2290
 390 PRINT I;"  ";J;"  ";K;TAB (30);TLST
 400 NEXT K @ NEXT J @ NEXT I
 410 TTGB=TTGB+DT(SOURCE)
 420 PRINT "DATA TRANSFER COMPLETED: TOTAL TIME GONE BY ";TTGB
 430 GOSUB 2310
 440 PRINT "NEXT TRANSMISSION REQUEST BY ";SAVESM @ SAVEIJK=SAVESM
 450 TTNR=SMALL
 460 IF TTNR<0 THEN 3480
 470 TTGB=TTGB+TTNR
 480 TTGBSLT=TTNR @ SAVESM2=0 @ SMALL=0 @ SMALL2=0
 481 ! PRINT OUT CURRENT ENTRY DATA - EITHER FROM FILE OR USER ENTERED
 490 PRINT "TOTAL TIME THAT WILL BE GONE BY WHEN ";SAVESM;" STARTS TO TRANSMIT "
:TTGB
3500 NUMDEST=0 @ PROB=0 @ SAVEATR=TTNR @ SAVESM=0
3505 ! ****************************************************************
3506 ! SOURCE DUMP ALGORITHM   LINES 3510 - 3590
3507 ! ****************************************************************
```

```
3510 FOR H=1 TO SNUM @ IF TIM(H)<= TTGB-WT(H) THEN 3520 ELSE 3590
520 WT(H)=TTGB @ J=INT (DTFL(H)/1000) @ K=INT ((DTFL(H)-J*1000)/100) @ L=INT ((
 TFL(H)-J*1000-K*100)/10) @ M=DTFL(H)-J*1000-K*100-L*10
 530 CP1=BYTES(H)*LT(J,K) @ CP2=BYTES(H)*LT(L,M)
 535 PRINTER IS 701 @ PRINT " " @ PRINT "SOURCE ";J;",";K;"    DUMPED TO DESTINA
 ION ";L;",";M;"  ";BYTES(H);" BYTES OF DATA"
 536 PRINT "TOTAL TIME GONE BY (BEFORE THE DUMP) ";TTGB
 540 IF CP1>CP2 THEN TL=CP1 ELSE TL=CP2
 550 TRANS=TSR+2000/CR+TL @ TTGB=TTGB+TRANS @ BBT=BBT+TRANS @ TGBSLT=TRANS
 555 PRINT "TOTAL TIME GONE BY (AFTER THE DUMP) ";TTGB @ PRINT " " @ IF PR=1 THE
  PRINTER IS 1
 556 IF PR=3 THEN 3557 ELSE 3560
 557 IF ITERNUM>= NUMITER-NUMCYCLES THEN 3560 ELSE PRINTER IS 1
 560 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
 570 ON I GOSUB 2210,2220,2230,2240,2250,2260,2270,2280,2290
 580 NEXT K @ NEXT J @ NEXT I
 590 NEXT H @ TTGBSLT=TTNR
 600 IF ITERNUM=NUMITER-1 THEN 3610 ELSE ITERNUM=ITERNUM+1 @ GOTO 2010
 610 PRINTER IS 701
 615 ! ***********************************************************************
 616 ! THE FINAL PRINTOUT GIVING A SUMMARY OF THE RUN
 617 ! LINES 3620 - 3830
 618 ! ***********************************************************************
 620 TOT=0 @ PRINT " " @ PRINT "SOURCE TO DESTINATION         # OF TIMES TRANSMIT
 ED"
 630 FOR I=1 TO NUMPROB @ PRINT PDS(I);TAB (30);TIMESTT(I) @ TOT=TOT+TIMESTT(I)
 640 NEXT I
 650 PRINT "TOTAL";TAB (30);TOT
 660 PRINT " " @ PRINT "PORT,DEVICE,SOURCE     LONGEST WAITING TIME   AVG WAITIN
  TIME   COLLISIONS"
 570 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
 580 ON I GOSUB 3720,3730,3740,3750,3760,3770,3780,3790,3800
 590 IF TN=0 THEN AVG=0 ELSE AVG=TNEG/TN
 700 PRINT I;"  ";J;"  ";K;TAB (25);MNEG;TAB (46);AVG;TAB (68);COLL
 710 NEXT K @ NEXT J @ NEXT I @ GOTO 3810
 720 MNEG=MNEG1(J,K) @ TNEG=TNG1(J,K) @ TN=TN1(J,K) @ COLL=COL1(J,K) @ RETURN
 730 MNEG=MNEG2(J,K) @ TNEG=TNG2(J,K) @ TN=TN2(J,K) @ COLL=COL2(J,K) @ RETURN
 740 MNEG=MNEG3(J,K) @ TNEG=TNG3(J,K) @ TN=TN3(J,K) @ COLL=COL3(J,K) @ RETURN
 750 MNEG=MNEG4(J,K) @ TNEG=TNG4(J,K) @ TN=TN4(J,K) @ COLL=COL4(J,K) @ RETURN
 760 MNEG=MNEG5(J,K) @ TNEG=TNG5(J,K) @ TN=TN5(J,K) @ COLL=COL5(J,K) @ RETURN
 770 MNEG=MNEG6(J,K) @ TNEG=TNG6(J,K) @ TN=TN6(J,K) @ COLL=COL6(J,K) @ RETURN
 780 MNEG=MNEG7(J,K) @ TNEG=TNG7(J,K) @ TN=TN7(J,K) @ COLL=COL7(J,K) @ RETURN
3790 MNEG=MNEG8(J,K) @ TNEG=TNG8(J,K) @ TN=TN8(J,K) @ COLL=COL8(J,K) @ RETURN
3800 MNEG=MNEG9(J,K) @ TNEG=TNG9(J,K) @ TN=TN9(J,K) @ COLL=COL9(J,K) @ RETURN
3810 PRINT "TOTAL TIME GONE BY = ";TTGB
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
~320 PRINT "BUS BUSY TIME = ":BBT
 330 END
 340 ! ****************************************************************
 345 ! PRINT OUT CURRENT DATA - EITHER FROM FILE OR ENTERED BY USER
 346 ! LINES 3850 - 4200
 347 ! ****************************************************************
 350 PRINTER IS 701
 360 PRINT "# OF HYFERCHANNEL PORTS  = ":HY @ PRINT " "
 370 FOR I=1 TO HY @ PRINT "# OF DEVICES FOR PORT":I:" = ":DV(I) @ NEXT I @ I=1
  PRINT " "
 380 FOR K=1 TO DV(I)
 390 PRINT "# OF SOURCES FOR PORT ":I;" DEVICE ":K;" = ":SR(I,K)
 900 NEXT K @ I=I+1 @ IF I<= HY THEN 3880 ELSE I=0 @ PRINT " "
 910 PRINT "PORT,DEVICE,SOURCE          AVERAGE ARRIVAL RATE        # OF BYTES ACCU
JLATED"
 920 I=I+1 @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
 930 ON I GOSUB 480,520,560,600,640,680,720,760,800
<340 PRINT I;"   ";J;"    ";K:TAB (35);LA;TAB (60);QQ
 950 NEXT K @ NEXT J
 960 IF I=HY THEN 3970 ELSE GOTO 3920
 970 PRINT " " @ PRINT " " @ FOR I=1 TO HY @ FOR J=1 TO DV(I)
 980 PRINT "PROPAGATION DISTANCE BETWEEN PORT ";I;" AND DEVICE ";J;" = ":D(I,J)
 990 NEXT J @ NEXT I @ PRINT " "
 000 FOR I=1 TO HY
 010 PRINT "WAIT TIME OF PORT ";I;" BEFORE TRANSMISSION REQUEST (IN EVENT OF A C
LLISION) ";W(I)
 020 NEXT I
 030 PRINT " " @ PRINT " " @ PRINT "CHANNEL SETUP AND RELEASE TIME = ";TSR @ PRI
T " "
 040 PRINT "CHANNEL DATA TRANSFER RATE = ":CR
 050 PRINT " " @ PRINT "PORT,DEVICE                    TRANSFER RATE           LOAD T
MES "
 060 FOR I=1 TO HY @ FOR J=1 TO DV(I)
 070 PRINT I;"    ";J;TAB (30);R(I,J);TAB (54);LT(I,J)
 080 NEXT J @ NEXT I
 090 PRINT " " @ PRINT "COMBINATIONS (IJK - LMN)  PROBABILITY  TIME TO TRANSMIT
 (IJK) BYTES (IJK - LMN)"
 100 FOR I=1 TO NUMPROB
 110 PRINT PDS(I);TAB (30);PRO(I);TAB (40);T(I)
 120 NEXT I
 130 PRINT " " @ PRINT "PORT,DEVICE.SOURCE     AVERAGE TIME TO NEXT TRANSMISSION
 REQUEST "
 140 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
4150 ON I GOSUB 480,520,560,600,640,680,720,760.800
```

```
4160 PRINT I;"  ";J,"  ";K:TAB (30);ATR
4170 NEXT K @ NEXT J @ NEXT I
180 PRINT " " @ PRINT "IJ-LM DATA FILE DUMP    BYTES TO BE DUMPED   TIME BETWEE
 DUMPS (IN SECONDS)"
190 FOR I=1 TO SNUM @ PRINT DTFL(I);TAB (30);BYTES(I);TAB (50);TIM(I)
200 NEXT I @ GOTO 1720
210 ! *******************************************************************
215 ! THE EDIT ROUTINE - SAVES CORRECTED DATA TO FILE
216 ! LINES 4220 - 4950
217 ! *******************************************************************
220 CLEAR @ DISP "       EDIT MENU" @ DISP " "
230 DISP "1  --  AVERAGE ARRIVAL RATES"
240 DISP "2  --  NUMBER OF BYTES ACCUMULATED"
250 DISP "3  --  PROPAGATION DISTANCES"
260 DISP "4  --  WAIT TIMES"
270 DISP "5  --  CHANNEL SETUP AND RELEASE TIME"
280 DISP "6  --  CHANNEL DATA TRANSFER RATE"
290 DISP "7  --  TRANSFER RATES"
300 DISP "8  --  PROBABILITY DATA"
310 DISP "9  --  SOURCE DATA DUMPS"
320 DISP "10 --  TERMINATE EDIT MODE"
330 DISP " " @ DISP "ENTER YOUR SELECTION ";@ INPUT EDT
340 IF EDT<1 OR EDT>10 THEN 4210
350 ON EDT GOTO 4360,4450,4540,4590,4630,4660,4690,4760,4850,4950
360 FOR I=1 TO HY @ CLEAR @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ CLEAR
370 ON I GOSUB 480,520,560,600,640,680,720,760,800
380 DISP "PORT ";I;" DEVICE ";J;" SOURCE ";K @ DISP @ DISP
390 LA(J,K)=LA @ QQ(J,K)=QQ
100 DISP "AVERAGE ARRIVAL RATE = ";LA
110 Q$=" " @ DISP " " @ DISP " " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR
ITER THE CORRECT VALUE ";@ INPUT Q$
120 IF Q$ <> "" THEN LA(J,K)=VAL (Q$)
130 NEXT K @ NEXT J @ ON I GOSUB 450,490,530,570,610,650,690,730,770
140 NEXT I @ GOTO 4210
150 FOR I=1 TO HY @ CLEAR @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ CLEAR
160 ON I GOSUB 480,520,560,600,640,680,720,760,800
70 DISP "PORT ";I;" DEVICE ";J;" SOURCE ";K @ DISP @ DISP
180 QQ(J,K)=QQ @ LA(J,K)=LA
90 DISP "NUMBER OF BYTES ACCUMULATED = ";QQ
00 Q$=" " @ DISP " " @ DISP " " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR
ITER THE CORRECT VALUE ";@ INPUT Q$
10 IF Q$ <> "" THEN QQ(J,K)=VAL (Q$)
4520 NEXT K @ NEXT J @ ON I GOSUB 450,490,530,570,610,650,690,730,770
4530 NEXT I @ GOTO 4840
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
540 CLEAR @ FOR I=1 TO HY @ FOR J=1 TO DV(I) @ CLEAR
550 DISP "PROPAGATION DISTANCE BETWEEN PORT ";I;" AND DEVICE ";J;" IN MICROSECO
DS = ";D(I.J) @ Q$=" "
560 DISP " " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VA
UE ";@ INPUT Q$
570 IF Q$ <> "" THEN D(I,J)=VAL (Q$)
580 NEXT J @ NEXT I @ GOTO 4210
590 CLEAR @ FOR I=1 TO HY @ CLEAR @ DISP "WAIT TIME OF PORT ";I;" BEFORE TRANSM
SSION REQUEST = ";W(I)
500 DISP " " @ Q$=" " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE C
RRECT VALUE ";@ INPUT Q$
510 IF Q$ <> "" THEN W(I)=VAL (Q$)
520 NEXT I @ GOTO 4210
530 CLEAR @ DISP "CHANNEL SETUP AND RELEASE TIME = ";TSR
540 Q$=" " @ DISP " " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE P
OPER VALUE ";@ INPUT Q$@ IF Q$ <> "" THEN TSR=VAL (Q$)
550 GOTO 4840
560 CLEAR @ DISP "CHANNEL DATA TRANSFER RATE = ";CR @ DISP " "
570 Q$=" " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENER THE PROPER VALUE
 @ INPUT Q$@ IF Q$ <> "" THEN CR=VAL (Q$)
580 GOTO 4840
590 CLEAR @ FOR I=1 TO HY @ FOR J=1 TO DV(I) @ CLEAR
'00 DISP "RATE AT WHICH DATA IS TRANSFERED FROM DEVICE ";J;" TO PORT ";I;" = ";
 I,J)
'10 Q$=" " @ DISP " " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE C
RECT VALUE ";@ INPUT Q$
 20 IF Q$ <> "" THEN R(I,J)=VAL (Q$)
'30 LT(I,J)=1/R(I,J)
 40 NEXT J @ NEXT I
 50 GOTO 4840
 60 CLEAR @ FOR I=1 TO NUMPROB
 70 CLEAR @ DISP "(PORT,DEVICE,SOURCE) TRANSMITS TO (PORT,DEVICE,SOURCE)  CODE
 ";PDS(I) @ DISP " "
 80 Q$=" " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VALU
 ";@ INPUT Q$
 90 IF Q$ <> "" THEN PDS(I)=VAL (Q$)
 00 DISP " " @ DISP "PROBABILITY = ";PRJ(I) @ DISP " "
 10 Q$=" " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VALU
 ";@ INPUT Q$
4820 IF Q$ <> "" THEN PRO(I)=VAL (Q$)
```

```
4830 NEXT I
4840 FOR I=1 TO NUMPROB @ GOSUB 1160 @ NEXT I @ GOTO 4210
4850 CLEAR @ FOR I=1 TO SNUM @ CLEAR @ DISP "(IJ - LM) SOURCE DATA FILE DUMPS TO
 DESTINATION LM. CODE = ";DTFL(I) @ DISP " "
4860 Q$=" " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VALU
E ";@ INPUT Q$
4870 IF Q$ <> "" THEN DTFL(I)=VAL (Q$)
4880 DISP " " @ DISP "# OF BYTES TO BE DUMPED ";BYTES(I) @ DISP " "
4890 Q$=" " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VALU
E ";@ INPUT Q$
4900 IF Q$ <> "" THEN BYTES(I)=VAL (Q$)
4910 DISP " " @ DISP "TIME INTERVAL BETWEEN DUMPS ";TIM(I) @ DISP " "
4920 Q$=" " @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VALU
E ";@ INPUT Q$
4930 IF Q$ <> "" THEN TIM(I)=VAL (Q$)
4940 NEXT I @ GOTO 4210
4950 GETDATA=888 @ ON AN GOTO 1350,1720,1350
```

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX II

PASCAL SIMULATIONS ALGORITHM PROGRAM

A listing of the PASCAL simulation algorithm program is presented in this appendix. The program contains comments, but is not considered to be user oriented at this point. This program is FORTRAN based, but requires software support for PASCAL in the host computer.

```
@ASG,A AUX-OUT.
FACILITY WARNING   100000000000

@ASG,A DESCRIP-FILE.
FACILITY WARNING   100000000000

@PAS,IS
Pascal (Ver 2.0ILR, Updated 09/24/81) - 07/01/83 16:39:06
      1    000000 --
      2    000000 --
      3    000000 --     (**************************************************************
      4    000000 --     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      5    000000 --     * *                                                          * *
      6    000000 --     *                                                              *
      7    000000 --     * *    TITLE:                                                * *
      8    000000 --     *        Simulation of a HYPERchannel Network:                 *
      9    000000 --     * *        A Model of the Huntsville Operations Support Center * *
     10    000000 --     *                                                              *
     11    000000 --     * *   AUTHOR:                                                  *
     12    000000 --     *       John D. Mauldin                                          *
     13    000000 --     * *                                                          * *
     14    000000 --     *       Mississippi State University                             *
     15    000000 --     * *   May, 1983                                                * *
     16    000000 --     *                                                              *
     17    000000 --     * *                                                          * *
     18    000000 --     *                                                              *
     19    000000 --     * *   ABSTRACT:                                                * *
     20    000000 --     *                                                              *
     21    000000 --     * *                                                          * *
     22    000000 --     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
     23    000000 --     **************************************************************)
     24    000000 --
     25    000000 --
     26    000000 --     program MOD1(INPUT,OUTPUT,TERMINAL,DESCRIP_FILE,AUX_OUT);
     27    000152 --
     28    000152 --                (*********************************************************)
     29    000152 --                (*********************************************************)
     30    000152 --                (*               D A T A   D E C L A R A T I O N S      *)
     31    000152 --                (*********************************************************)
     32    000152 --
     33    000152 --
     34    000152 --       const
     35    000152 --         MAX_NUM_SOURCES  = 3;
     36    000152 --         MAX_NUM_ADAPTERS = 4;
     37    000152 --         NUM_OF_ADAPTERS  = 4;
     38    000152 --         LPP              = 80;
     39    000152 --         TNUM_DEV         = 11;
     40    000152 --         TRUNK_OVERHEAD   = 0.000025 ; (* Trunk set-up and release time *)
     41    000152 --         TRUNK_RATE       = 50E6 ;  (* Trunk tx rate in bits/second  *)
     42    000152 --         ONE              = 1;
     43    000152 --         ZERO             = 0;
     44    000152 --         COLL_EPS         = 0.001;  (* Increment to determine collisions *)
     45    000152 --
     46    000152 --       type
     47    000152 --         SOURCE_DESCRIPTIONS = record
     48    000152 --           TX_RATE    : real ; (* Avg number of bytes/sec *)
     49    000152 --         end;
     50    000152 --
     51    000152 --
     52    000152 --         DEVICE_RECORD  = record
     53    000152 --            OPEN             : boolean;    (* TRUE if device attached to port *)
     54    000152 --            DEV_NUM          : integer;    (* Device number *)
     55    000152 --            BUFFER_SIZE      : integer;    (* Number of bits accumulated from a
     56    000152 --                                            off-net source before the device requests
     57    000152 --                                            a trunk transmission *)
     58    000152 --            NUM_OF_SOURCES   : integer;
     59    000152 --            SOURCE           : array[1..MAX_NUM_SOURCES] of SOURCE_DESCRIPTIONS;
     60    000152 --            TFER_RATE        : real;       (* I/O port transfer rate from dev to adapt in bytes/sec *)
     61    000152 --            LOAD_TIME        : real;       (* inverse of TFER_RATE = time for dev to load adapt buffer *)
     62    000152 --            NEXT_TX          : real;       (* time until next transmission*)
     63    000152 --            LAST_TRUNK_TX    : real;
     64    000152 --            TX_INTRVL        : real;       (* time between tx requests based on
     65    000152 --                                            aggregate source trans. rates from offnet sources *)
     66    000152 --            TX_CT            : integer;    (* Device transmission tally *)
     67    000152 --            WAIT_CT          : integer;    (* Device waits tally *)
     68    000152 --            COLL_CT          : integer;    (* Device collision tally *)
     69    000152 --            RX_CT            : integer;    (* Device receiving tally *)
     70    000152 --            TX_TIME          : real;       (* Time device spent in transmissions *)
```

```
000152 --        WAIT_TIME          : real;       (* Time device spent in wait delays *)
000152 --        COLL_TIME          : real;       (* Time device spent in coll. waits *)
000152 --        RX_TIME            : real;       (* Time device spent receiving trunk tx *)
000152 --      end;
000152 --
000152 --
000152 --      ADAPTER_RECORDS = record
000152 --        DEVICE          : array[1..4] of DEVICE_RECORDS;
000152 --        PROP_DIST_TO    : array[1..NUM_OF_ADAPTERS] of real;  (* Number of seconds
000152 --                                      required to transmit from one adapter
000152 --                                      to each other adapter on the net *)
000152 --        PRIORITY_DELAY       : real;    (* Number of seconds of assigned wait time in
000152 --                                 event of wait during transmission *)
000152 --        END_DELAY       : real;   (* Wait time if wait flip flop enabled *)
000152 --        ATX_CT          : integer;(* Tally of adapter transmissions *)
000152 --        AWAIT_CT        : integer;(* Tally of adapter waits     *)
000152 --        ACOLL_CT        : integer;(* Tally of adapter collisions  *)
000152 --        ATX_TIME        : real;   (* Time adapter spent transmitting *)
000152 --        AWAIT_TIME      : real;   (* Time adapter spent in wait delays *)
000152 --        ACOLL_TIME      : real;   (* Time adapter spent in collisions *)
000152 --      end;
000152 --
000152 --
000152 --      TEXTFILE = file of CHAR;
000152 --
000152 --      MODE_TYPE  =  (INTERACTIVE, FILE_INPUT);
000152 --
000152 --      CLOCK_COND =  (NORMAL,COLLISION,OTHER);
000152 --
000152 --
000152 --    var
000152 --      ADAPTER                     : array[1..NUM_OF_ADAPTERS] of ADAPTER_RECORDS;
000732 --      PR                          :  array[1..TNUM_DEV,1..TNUM_DEV] OF real;  (* Probability
000732 --                                              of transmission matrix *)
001123 --
001123 --      BITS_TX,
001123 --      TX_TALLY,TOTAL_ATTEMPTS,
001123 --      MAX_TX,
001123 --      WAIT_TALLY,
001123 --      COLLISION_TALLY,
001123 --      RETRY_TALLY,
001123 --      I,J,K,D_NUM                 : integer;
001136 --      MAX_TIME,                               (*  Run time for prog. in simulation secs *)
001136 --      CURRENT_TIME,                           (* System clock recording total elapsed time *)
001136 --      TIME_ACTIVE,
001136 --      LAST_TRUNK_TX               : real;     (* Time of last successful trunk transmission *)
001142 --      U,IU                        : integer;     (* Seed for random number generator *)
001144 --      RESP                        : packed array[1..80] of char;
001170 --      DESCRIP_FILE,AUX_OUT        : TEXTFILE;
003170 --      INPUT_MODE                  : MODE_TYPE;
003171 --      CONDITION                   : array[CLOCK_COND] of CLOCK_COND;
003174 --      COND1                       : CLOCK_COND;
003175 --      TMITTR,RCVR                 : DEVICE_RECORD;
003245 --      PRINT_ALL                   : boolean;
003246 --      LCT,PCT,JK                  : integer;
003251 --
003251 --
003251 --              (*********************************************************)
003251 --              (*********************************************************)
003251 --              (*        P R O C E D U R E     D E F I N I T I O N S     *)
003251 --              (********************************************************)
003251 --
003251 --
003251 --      (**************************************************)
003251 --      (*                                                *)
003251 -- A        function UNIFORM(ZERO,ONE:real; var U:INTEGER):real;
000011 --            extern;
000011 --      (*   Calculates a pseudorandom number in range     *)
000011 --      (*   0<num<1.                                      *)
000011 --      (*                                                *)
000011 --      (**************************************************)
000011 --
000011 --
000011 --      (*****************************************************)
000011 --      (*
000011 -- A        function TIME_TO_NEXT_TX(DEVICE: DEVICE_RECORD) :real;
000033 --      (*     Calculates the remaining time until device      *)
000033 --      (*     requests a transmission.                        *)
000033 --      (*****************************************************)
000033 --
000033 --
000033 --
```

```
000033 0- A      begin
000000 --          with DEVICE do
000013 --          TIME_TO_NEXT_TX:=(CURRENT_TIME - LAST_TRUNK_TX) - TX_INTRVL;
000020 -0 A      end;
000022 --
000022 --
000022 --        (*************************************************)
000022 --        (*                                               *)
000022 -- A          procedure PRINT_WAIT_STATS(TMTTR: DEVICE_RECORD);
000033 --        (*       Prints statistics on successful trunk tx.  *)
000033 --        (*                                               *)
000033 --        (*************************************************)
000033 --
000033 --
000033 --          var
000033 --            W_ADAPT,
000033 --            W_DEV           : integer;
000035 --
000035 --
000035 0- A        begin
000022 --            W_DEV:= TMTTR.DEV_NUM mod 10;
000041 --            W_ADAPT:= (TMTTR.DEV_NUM mod 100) div 10;
000050 --
000050 --
000050 --            writeln(OUTPUT);
000053 --            writeln(OUTPUT,'** WAIT CONDITION ENCOUNTERED **');
000062 --            writeln(OUTPUT,'      Time:  ',current_time:12:7);
000075 --            writeln(' ':3,'Transmitter:  adapter  ',W_ADAPT:3);
000112 --            writeln(' ':17,              'device   ',W_DEV:3);
000127 --
000127 -0 A        end;  (* PRINT TX STATS *)
000131 --
000131 --
000131 --        (*************************************************)
000131 --        (*                                               *)
000131 -- A          procedure PRINT_TX_STATS(TMTTR,RCVR: DEVICE_RECORD);
000060 --        (*       Prints statistics on successful trunk tx.  *)
000060 --        (*                                               *)
000060 --        (*************************************************)
000060 --
000060 --
000060 --          var
000060 --            T_ADAPT,R_ADAPT,
000060 --            T_DEV,R_DEV              : integer;
000064 --
000064 --
000064 0- A        begin
000131 --            T_DEV:= TMTTR.DEV_NUM mod 10;
000155 --            T_ADAPT:= (TMTTR.DEV_NUM mod 100) div 10;
000164 --
000164 --            R_DEV := RCVR.DEV_NUM mod 10;
000171 --            R_ADAPT:= (RCVR.DEV_NUM mod 100) div 10;
000200 --
000200 --            writeln(OUTPUT);
000203 --            writeln(OUTPUT,'** SUCCESSFUL TRANSMISSION **');
000212 --            writeln(OUTPUT,'      Time:  ',current_time:12:7);
000225 --            writeln(' ':3,'Transmitter:  adapter  ',T_ADAPT:3);
000242 --            writeln(' ':17,              'device   ',T_DEV:3);
000257 --            writeln(' ':3,'Reciever    :  adapter  ',R_ADAPT:3);
000274 --            writeln(' ':17,              'device   ',R_DEV:3);
000311 --
000311 -0 A        end;  (* PRINT TX STATS *)
000313 --
000313 --
000313 --        (*************************************************)
000313 --        (*                                               *)
000313 -- A          procedure PRINT_COLLISION_STATS(TMTTR,RCVR: DEVICE_RECORD);
000060 --        (*       Prints stats on event of collision.        *)
000060 --        (*                                               *)
000060 --        (*************************************************)
000060 --
000060 --
000060 --          var
000060 --            T_ADAPT,R_ADAPT,
000060 --            T_DEV,R_DEV              : integer;
000064 --
000064 --
000064 0- A        begin
000313 --            T_DEV:= TMTTR.DEV_NUM mod 10;
000337 --            T_ADAPT:= (TMTTR.DEV_NUM mod 100) div 10;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
000346 --
000346 --            R_DEV := RCVR.DEV_NUM mod 10;
000353 --            R_ADAPT:= (RCVR.DEV_NUM mod 100) div 10;
000362 --
000362 --            writeln(OUTPUT);
000365 --            writeln(OUTPUT,'** UNSUCCESSFUL TRANSMISSION -- COLLISION OCCURRED **');
000374 --            writeln(OUTPUT,'      Time:  ',current_time:12:7);
000407 --            writeln(' ':3,'Transmitter:  adapter  ',T_ADAPT:3);
000424 --            writeln(' ':17,'              device   ',T_DEV:3);
000441 --            writeln(' ':3,'Reciever    :  adapter  ',R_ADAPT:3);
000456 --            writeln(' ':17,'              device   ',R_DEV:3);
000473 --
000473 -0 A        end;  (* PRINT COLLISION STATS *)
000475 --
000475 --          (************************************************************)
000475 --          (*                                                          *)
000475 -- A            procedure NEWPAGE;
000006 --          (*      Spaces output page of AUX_OUT to top of new page.  *)
000006 --          (*                                                          *)
000006 --          (************************************************************)
000006 --
000006 0- A          begin
000475 --
000475 --              PCT:=PCT+1;
000506 --               LCT:=LPP-LCT+1;
000513 --               for JK:=1 to LCT do
000524 --                 writeln(AUX_OUT):
000533 --               LCT:=0;
000535 --               writeln(AUX_OUT,' ':120,'page ',PCT:2);
000552 --               LCT:=LCT+1;
000556 -0 A          end;  (* procedure NEWPAGE *)
000560 --
000560 --
000560 --          (*********************************************************)
000560 --          (*                                                       *)
000560 -- A           procedure PRINT_NETWORK_STATS;
000006 --          (*   Prints cumulative statistics of network at end     *)
000006 --          (*   of simulation.                                     *)
000006 --          (*                                                       *)
000006 --          (*********************************************************)
000006 --
000006 --        var
000006 --         PER_CENT_ACTIVE    : real;
000007 --
000007 0- A       begin
000560 --            NEWPAGE;
000571 --             PER_CENT_ACTIVE:=TIME_ACTIVE/CURRENT_TIME*100.0;
000576 --            writeln(AUX_OUT,' ':40,'*** END OF RUN NETWORK STATISTICS ***');
000610 --             writeln(AUX_OUT);
000613 --            writeln(AUX_OUT,' ':40,' Current time: ',CURRENT_TIME:12:4,' secs');
000635 --            writeln(AUX_OUT);
000640 --            writeln(AUX_OUT,' ':40,' Successful transmissions: ',TX_TALLY:5);
000655 --            writeln(AUX_OUT,' ':40,' Collisions               : ',COLLISION_TALLY:5);
000672 --            writeln(AUX_OUT,' ':40,' Waits                    : ',WAIT_TALLY:5);
000707 --            writeln(AUX_OUT,' ':40,' Total attempts           : ',TOTAL_ATTEMPTS:5);
000724 --            writeln(AUX_OUT);
000727 --            writeln(AUX_OUT,' ':40,' Total time active        : ',TIME_ACTIVE:10:7,' secs');
000751 --            writeln(AUX_OUT,' ':40,' % Total time active      : ',PER_CENT_ACTIVE:6:2,' %');
000771 --            writeln(AUX_OUT);
000774 --            writeln(AUX_OUT,' ':40,' Total Bytes transmitted : ',BITS_TX DIV 8:12);
001014 --            writeln(AUX_OUT);
001017 --            writeln(AUX_OUT,' ':40,' Seed for RNG             : ',IU:7);
001034 --            LCT:=LCT+16;
001040 --
001040 --            writeln(OUTPUT);writeln(OUTPUT);
001045 --            writeln(OUTPUT,'          *** END OF RUN ***');
001054 --            writeln(OUTPUT,'    Current time: ',CURRENT_TIME:12:4,' secs');
001073 --            writeln(OUTPUT);
001076 --            writeln(OUTPUT,'    Successful transmissions: ',TX_TALLY:5);
001110 --            writeln(OUTPUT,'    Collisions               : ',COLLISION_TALLY:5);
001122 --            writeln(OUTPUT,'    Waits                    : ',WAIT_TALLY:5);
001134 --            writeln(OUTPUT,'    Total attempts           : ',TOTAL_ATTEMPTS:5);
001146 --            writeln(OUTPUT);
001151 --            writeln(OUTPUT,' ':3,' Total time active        : ',TIME_ACTIVE:10:4,' secs');
001173 --            writeln(OUTPUT,' ':3,' % Total time active      : ',PER_CENT_ACTIVE:6:2,' %');
001213 --            writeln(OUTPUT);
001216 --            writeln(OUTPUT,' ':3,' Total Bytes transmitted : ',BITS_TX DIV 8:12);
001236 --            writeln(OUTPUT);
001241 --            writeln(OUTPUT,' ':3,' Seed for RNG             : ',IU:7);
001256 --
```

```
001256 --
001256 -0 A      end;  (* PRINT NETWORK STATS *)
001260 --
001260 --        (***************************************************************)
001260 --        (*                                                             *)
001260 -- A          function TRANSFER_TIME(SENDER,RECEIVER :
000006 --                                          DEVICE_RECORD): real;
000060 --        (*             Calculates time required to transmit a message  *)
000060 --        (*             from a sender to a receiver.                    *)
000060 --        (*     Trunk overhead time includes the fixed delay for each   *)
000060 --        (*     adapter which is length of time required by a sending   *)
000060 --        (*     device to receive a response frame.                     *)
000060 --        (***************************************************************)
000060 --
000060 --          var
000060 --            T,SLOWER_RATE:  real;
000062 --            PCKT_CT      :  integer;
000063 --
000063 --
000063 0- A      begin
001260 --
001260 --          SLOWER_RATE:= SENDER.TFER_RATE;
001302 --          if RECEIVER.TFER_RATE < SLOWER_RATE then
001310 --             SLOWER_RATE:= RECEIVER.TFER_RATE;
001313 --
001313 --          PCKT_CT:= round(SENDER.BUFFER_SIZE/2048);
001323 --
001323 1-         case PCKT_CT of
001326 --            0:  T:=TRUNK_OVERHEAD;
001332 --
001332 --            1:  T:= 2048/TRUNK_RATE + TRUNK_OVERHEAD;
001340 --
001340 --            otherwise:  T:= 2048/SENDER.TFER_RATE + 2048/TRUNK_RATE +
001346 --                            ((PCKT_CT-2)*2048)/SLOWER_RATE + TRUNK_OVERHEAD
001356 -1          end;
001371 --
001371 --          TRANSFER_TIME:=T;
001374 --
001374 -0 A      end;
001376 --
001376 --
001376 --       (****************************************************)
001376 --    (*                                                     *)
001376 -- A        procedure UPDATE_CLOCKS(COND:CLOCK_COND);
000007 --        (*    Updates both the system clocks and device     *)
000007 --        (*    clocks whenever a system action has           *)
000007 --        (*    occurred.                                     *)
000007 --        (*                                                  *)
000007 --        (****************************************************)
000007 --
000007 --
000007 -          var
000007 --           TA,TD,RA,RD,
000007 --           I,J        : integer;
000015 --           DELAY_TIME,
000015 --           TIME,T1    : real;
000020 --
000020 0- A      begin
001376 --          (* A is transmitter adapter no., D is devive no. *)
001376 --
001376 --          TA:= (TMITTR.DEV_NUM mod 100) div 10;
001417 --          RA:= (RCVR.DEV_NUM mod 100) div 10;
001426 --
001426 --          TD:= TMITTR.DEV_NUM mod 10;
001433 --          RD:= RCVR.DEV_NUM mod 10;
001440 --
001440 --
001440 1-           case COND of
001443 2-              NORMAL      : begin
001444 --                   TIME:= TRANSFER_TIME(TMITTR,RCVR);
001454 --
001454 --                   BITS_TX:=BITS_TX+ TMITTR.BUFFER_SIZE;
001460 --
001460 --
001460 --                    (* Update system clock *)
001460 --
001460 --                   CURRENT_TIME:= CURRENT_TIME+TIME;
001464 --
001464 --                    (* Update each device clock *)
001464 --
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
001464 --               for I:= 1 to NUM_OF_ADAPTERS do
001472 --                for J:=1 to 4 do
001500 --                 with ADAPTER[I].DEVICE[J] do
001520 --                  if (not OPEN) and (NOT((TA=I) and (TD=J))) then
001533 3-                    begin
001533 --                     if NEXT_TX <= CURRENT_TIME then
001542 4-                       begin
001542 --                         DELAY_TIME:=ADAPTER[I].PRIORITY_DELAY + ADAPTER[I].END_DELAY;
001564 --                         NEXT_TX:=CURRENT_TIME +  DEL:Y_TIME;
001571 --                         WAIT_TALLY:= WAIT_TALLY + 1;
001575 --                         WAIT_CT:=WAIT_CT+T;
001603 --                         WAIT_TIME:=WAIT_TIME+DELAY_TIME+TIME;
001612 --
001612 --                         if PRINT_ALL then
001614 --                           PRINT_WAIT_STATS(ADAPTER[I].DEVICE[J]);
001637 -4                      end; (* NEXT_TX <= 0 *)
001637 -3                    end;
001650 --
001650 --                  (* Reset transmitter clock *)
001650 --                with ADAPTER[TA].DEVICE[TD] do
001670 3-                 begin
001670 --                  NEXT_TX:= CURRENT_TIME+ TX_INTRVL;
001676 --                  LAST_TRUNK_TX:=CURRENT_TIME;
001702 --                  TX_CT:=TX_CT + 1;
001710 --                  TX_TIME:=TX_TIME + TIME;
001716 -3               end; (* with ADAPTER[TA].DEVICE[TD] *)
001716 --
001716 --               with ADAPTER[RA].DEVICE[RD] do
001736 3-                 begin
001736 --                  RX_CT:=RX_CT + 1;
001744 --                  RX_TIME:=RX_TIME+TIME;
001752 -3               end; (* with ADAPTER[RA].DEVICE[RD] *)
001752 --
001752 --               TIME_ACTIVE:=TIME_ACTIVE+TRANSFER_TIME(TMITTR,RCVR);
001763 --
001763 --
001763 -2             end;  (* NORMAL case *)
001764 --
001764 2-        COLLISION  : begin
001765 --
001765 --               with ADAPTER[TA].DEVICE[TD] do
002005 3-                 begin
002005 --                  T1:= COLL_EPS + (UNIFORM(ZERO,ONE,U)/(1.0/COLL_EPS));
002023 --                  NEXT_TX:=NEXT_TX+ T1;
002031 --                  COLL_CT:=COLL_CT+1;
002037 --                  COLL_TIME:=COLL_TIME+ T1;
002045 -3                end;  (* with ADAPTER[A].DEVICE[D] *)
002045 -2             end;    (* COLLISION case *)
002046 --
002046 2-        OTHER       : begin
002047 --
002047 -2             end;    (* OTHER case *)
002050 -1      end;  (*  case *)
002061 --
002061 --
002061 -0 A end;  (* UPDATE CLOCKS *)
002063 --
002063 --
002063 --
002063 --
002063 --      (************************************************)
002063 --      (*                                              *)
002063 -- A       function TX_INTERVAL(DEVICE : DEVICE_RECORD) :real;
000033 --      (*     Calculates the time interval between requests *)
000033 --      (*     for trunk transmissions.  Based on the amount *)
000033 --      (*     of data (rate) received by a device from an   *)
000033 --      (*     off-net sources and the size of the device    *)
000033 --      (*     buffer.                                      *)
000033 --      (************************************************)
000033 --
000033 --
000033 --      var
000033 --        I              : integer;
000034 --        AGGREGATE_RATE: real;
000035 --
000035 0- A     begin
002063 --
002063 --        AGGREGATE_RATE := 0.00;
002077 --
002077 --        for I := 1 to DEVICE.NUM_OF_SOURCES do
```

```
002110 1-          begin
002110 --            AGGREGATE_RATE := AGGREGATE_RATE + DEVICE.SOURCE[I].TX_RATE
002116 -1          end;
002127 --
002127 --        if AGGREGATE_RATE <> 0.0 THEN
002133 --          TX_INTERVAL := DEVICE.BUFFER_SIZE/AGGREGATE_RATE
002133 --        else
002142 --          TX_INTERVAL := 1E7;
002145 --
002145 -0 A   end;
002147 --
002147 --
002147 --
002147 --      (**************************************************************)
002147 --      (*                                                            *)
002147 -- A       procedure FIND_NEXT(var TMITTER:DEVICE_RECORD);
000007 --      (*     Determines the next transmitter by examining         *)
000007 --      (*     each device's time until next transmission.          *)
000007 --      (*     Next tmittr is device with shortest time until       *)
000007 --      (*     next transmission.                                   *)
000007 --      (*                                                          *)
000007 --      (************************************************************)
000007 --
000007 --        var
000007 --          I,J        : integer;
000011 --
000011 --
000011 0- A     begin
002147 --
002147 --        TMITTER.NEXT_TX:=1000.0;
002161 --
002161 --        for I:=1 to NUM_OF_ADAPTERS do
002167 --          for J:=1 to 4 do
002175 --            with ADAPTER[I].DEVICE[J] do
002215 --              if not OPEN then
002221 --                if NEXT_TX < TMITTER.NEXT_TX then
002231 --                  TMITTER:=ADAPTER[I].DEVICE[J];
002265 --
002265 -0 A     end;  (* FIND NEXT TRANSMITTER *)
002267 --
002267 --
002267 --
002267 --      (***********************************************************)
002267 --      (*                                                         *)
002267 -- A       procedure PICK_A(var RECEIVER:DEVICE_RECORD;SENDER:DEVICE_RECORD);
000034 --      (*     Picks an eligible receiver for the current          *)
000034 --      (*     transmitter based on the normalized trnsmission     *)
000034 --      (*     probability matrix.                                 *)
000034 --      (*                                                         *)
000034 --      (***********************************************************)
000034 --
000034 --
000034 --      var
000034 --        I,J,K,L,
000034 --        A,D       : integer;
000042 --        FOUND     : boolean;
000043 --        PROB      : real;
000044 --
000044 0- A     begin
002267 --
002267 --        PROB := UNIFORM(ZERO,ONE,U);
002314 --
002314 --        I:= SENDER.DEV_NUM div 100;
002321 --        J:= 1;
002323 --        while (PR[I,J] < PROB) and (J < D_NUM) do
002350 --          J:=J+1;
002356 --
002356 --          (* J is now receiver device number                    *)
002356 --          (* Find adapter and device num for dev num J          *)
002356 --
002356 --            A:=1;  K:=1;
002361 --            D:=1;  L:=1;
002364 --            FOUND:=false;
002366 --
002366 --            while (K<= NUM_OF_ADAPTERS) and (not FOUND) do
002374 1-              begin
002374 --                while (L<= 4 ) and (not FOUND) do
002402 2-                  begin
002402 --                    with ADAPTER[K].DEVICE[L] do
002422 3-                      begin
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
002422 --                        if not OPEN then
002426 --                          if DEV_NUM div 100 = J then
002435 4-                           begin
002435 --                             FOUND:=true;
002437 --                             A:=K;
002442 --                             D:=L;
002445 -4                         end;  (* both ifs *)
002445 --                        L:=L+1;
002451 -3                       end; (* with *)
002451 -2                   end; (* while L<=4 and not FOUND *);
002454 --                  K:=K+1;
002460 --                  L:=1;
002462 -1                end; (* while K<= num adapters and not FOUND *)
002465 --
002465 --           RECEIVER:=ADAPTER[A].DEVICE[D];
002511 --
002511 -0 A        end;  (* PICK A RECEIVER *)
002513 --
002513 --
002513 --
002513 --         (*****************************************************)
002513 --         (*                                                   *)
002513 -- A           function A_COLLISION : boolean;
000006 --         (*   determines the chance of a collision occurring   *)
000006 --         (*   for a given transmission attempt.  Collision     *)
000006 --         (*   occurs if two transmitters attempt to transmit  *)
000006 --         (*   within 0.01 seconds of each other.              *)
000006 --         (*                                                   *)
000006 --         (*****************************************************)
000006 --
000006 --           var
000006 --             I,J   : integer;
000010 --
000010 --
000010 0- A        begin
002513 --
002513 --           A_COLLISION := false;
002522 --
002522 --           for I:=1 to NUM_OF_ADAPTERS do
002530 --             for J:= 1 to 4 do
002536 --               with ADAPTER[I].DEVICE[J] do
002556 --                 if not OPEN   then
002562 --                   if TMITTR.DEV_NUM <> DEV_NUM then
002567 --                     if abs(TMITTR.NEXT_TX-NEXT_TX) < COLL_EPS then
002600 --                       A_COLLISION:= true;
002612 --
002612 -0 A        end;  (* COLLISION DETECTION *)
002614 --
002614 --
002614 --          (*****************************************************)
002614 --          (*                                                   *)
002614 -- A              procedure ACTIVITY_SUMMARY;
000006 --          (*                                                   *)
000006 --          (*****************************************************)
000006 --
000006 --
000006 --           var
000006 --             I,J            : integer;
000010 --             PC_WAIT_DEV,
000010 --             PC_ACT_TR,
000010 --             PC_TX_TR,
000010 --             PC_TX_DEV,
000010 --             PC_WAIT_TR,
000010 --             WAIT_PC_DEV,
000010 --             PC_COLL_TR,
000010 --             PC_COLL_DEV,
000010 --             DEV_ACTIVE_TIME    : real;
000021 --
000021 0- A        begin
002614 --
002614 --             writeln(AUX_OUT);
002624 --             writeln(AUX_OUT);
002627 --             writeln(AUX_OUT);
002632 --             writeln(AUX_OUT);
002635 --             writeln(AUX_OUT,' ':45,'DEVICE ACTIVITY SUMMARIES');
002647 --             writeln(AUX_OUT,' ':52,'(SECONDS)');
002661 --             writeln(AUX_OUT);
002664 --             write(AUX_OUT,' ':2,'ADP DEV');
002675 --             write(AUX_OUT,' ':3,'TIME ',' ':9,'TIME',' ':10);
002720 --             write(AUX_OUT,'TIME IN ',' ':7,'TIME ',' ':10,'TIME');
```

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

```
002744 --          write(AUX_OUT,' ':10,'TRANSMISSION',' ':2,'RECEPTION',' ':4,'WAIT',' ':3);
002776 --          writeln(AUX_OUT,'COLLISION');
003005 --
003005 --          write(AUX_OUT,'  #   #',' ':2,'TRANSMITTING',' ':2,'WAITING');
003031 --          write(AUX_OUT,' ':7,'COLLISIONS',' ':4,'RECEIVING',' ':5);
003054 --          write(AUX_OUT,'ACTIVE',' ':12,'COUNT',' ':8,'COUNT',' ':5);
003103 --          writeln(AUX_OUT,'COUNT',' ':4,'COUNT');
003121 --
003121 --          writeln(AUX_OUT);
003124 --            LCT:=LCT+9;
003130 --
003130 --          for I:= 1 to NUM_OF_ADAPTERS do
003136 --            for J:= 1 to 4 do
003144 --             with ADAPTER[I].DEVICE[J] do
003164 --              if not OPEN then
003170 1-             begin
003170 --               DEV_ACTIVE_TIME:=TX_TIME+WAIT_TIME+COLL_TIME+RX_TIME;
003202 --               write(AUX_OUT,I:4,J:4,TX_TIME:12:4,WAIT_TIME:13:4);
003224 --               write(AUX_OUT,COLL_TIME:15:4,RX_TIME:13:4,DEV_ACTIVE_TIME:13:4);
003244 --               write(AUX_OUT,TX_CT:13,RX_CT:14,WAIT_CT:8);
003262 --               writeln(AUX_OUT,COLL_CT:9);
003271 --               writeln(AUX_OUT);
003274 --               LCT:=LCT+2;
003300 -1             end; (* if not OPEN *)
003311 --
003311 --          (* Calculate and print percent summaries *)
003311 --
003311 --          writeln(AUX_OUT);writeln(AUX_OUT);
003316 --          writeln(AUX_OUT);
003321 --          writeln(AUX_OUT,' ':45,'DEVICE ACTIVITY SUMMARIES');
003333 --          writeln(AUX_OUT,' ':53,'(PERCENT)');
003345 --          writeln(AUX_OUT);
003350 --          write(AUX_OUT,' ':2,'ADP DEV');
003361 --          write(AUX_OUT,' ':5,'% TIME',' ':8,'% TIME',' ':8);
003404 --          write(AUX_OUT,'% TIME',' ':8,'% TIME',' ':8,'% TIME');
003430 --          writeln(AUX_OUT,' ':6,'% TIME IN',' ':5,'% TIME IN');
003451 --
003451 --          write(AUX_OUT,' ':14,'ACTIVE',' ':5,'TRANSMITTING',' ':2);
003474 --          write(AUX_OUT,' ':2,'TRANSMITTING',' ':3,'WAITING',' ':7);
003517 --          write(AUX_OUT,'WAITING',' ':6,'COLLISIONS',' ':3);
003537 --          writeln(AUX_OUT,'COLLISIONS');
003546 --          write(AUX_OUT,' ':13,'(TRUNK)',' ':7,'(TRUNK)',' ':7);
003571 --          write(AUX_OUT,'(DEVICE)',' ':6,'(TRUNK)',' ':7);
003611 --          write(AUX_OUT,'(DEVICE)',' ':7,'(TRUNK)',' ':6);
003631 --          writeln(AUX_OUT,'(DEVICE)');
003640 --
003640 --          LCT:=LCT+7;
003644 --
003644 --          for I:=1 to NUM_OF_ADAPTERS do
003652 --            for J:=1 to 4 do
003660 --              with ADAPTER[I].DEVICE[J] do
003700 --                if not OPEN then
003704 1-                begin
003704 --                  DEV_ACTIVE_TIME:=TX_TIME+WAIT_TIME+COLL_TIME+RX_TIME;
003716 --                   if DEV_ACTIVE_TIME > 0.0 then
003723 2-                     begin
003723 --                         PC_ACT_TR:=DEV_ACTIVE_TIME/TIME_ACTIVE*100.0;
003730 --                         PC_TX_TR:=TX_TIME/TIME_ACTIVE*100.0;
003736 --                         PC_WAIT_TR:=DAIT_TIME/TIME_ACTIVE*100.0;
003744 --                         PC_COLL_TR:=COLL_TIME/TIME_ACTIVE*100.0;
003752 --                         if (TX_CT = 0) or (TX_INTRVL = 0.0) then
003762 --                           PC_TX_DEV:=0
003762 --                          else
003766 --                           PC_TX_DEV:=TX_TIME/TX_CT/TX_INTRVL*100.0;
004001 --                         if (GAIT_CT = 0) or (TX_INTRVL = 0.0) then
004011 --                           PC_WAIT_DEV:=0
004011 --                          else
004015 --                           PC_WAIT_DEV:=WAIT_TIME/WAIT_CT/TX_INTRVL*100.0;
004030 --                         if (COLL_CT = 0) or (TX_INTRVL = 0.0) then
004040 --                           PC_COLL_DEV:=0
004040 --                          else
004044 --                           PC_COLL_DEV:=COLL_TIME/COLL_CT/TX_INTRVL*100.0;
004057 -2                      end
004057 --                    else
004061 2-                      begin
004061 --                        PC_TX_DEV:=0.0;
004063 --                        PC_WAIT_DEV:=0.0;
004065 --                        PC_COLL_DEV:=0.0;
004067 --                        PC_ACT_TR:=0.0;
004071 -2                    end; (* if DEV_ACT_TIME =0.0 *)
```

```
004071 --
004071 --                        write(AUX_OUT,I:4,J:4,PC_ACT_TR:11:2,PC_TX_TR:14:2);
004111 --                        write(AUX_OUT,PC_TX_DEV:14:2,PC_WAIT_TR:14:2,PC_WAIT_DEV:14:2);
004127 --                        write(AUX_OUT,PC_COLL_TR:14:2);
004135 --                        writeln(AUX_OUT,PC_COLL_DEV:14:2);
004144 --
004144 --                        writeln(AUX_OUT);
004147 --
004147 --                        LCT:=LCT+2;
004153 -1                       end;  (* not OPEN *)
004164 --
004154 --
004164 --           NEWPAGE;
004170 --
004170 -0 A        end;  (* procedure ACTIVITY SUMMARY *)
004172 --
004172 --
004172 --    (*****************************************************)
004172 --    (*                                                   *)
004172 -- A    procedure CHARACTERIZE_NETWORK;
000006 --    (*        prompts for network descriptive info.      *)
000006 --    (*****************************************************)
000006 --
000006 --      var
000006 --        I,J    :integer;
000010 --
000010 --
000010 --        (*****************************************************)
000010 --        (*                                                   *)
000010 -- B          procedure INITIALIZE_PROB_MATRIX;
000006 --        (*    Initializes matrix containing the probability  *)
000006 --        (*    that any device will request a transmission to *)
000006 --        (*    any other device.  Matrix is a 2-D array .     *)
000006 --        (*    Row references refer to sender and columns     *)
000006 --        (*    refer to receiver.  Each device is assigned a  *)
000006 --        (*    unique device number upon  program startup     *)
000006 --        (*    and these numbers should be used to obtain     *)
000006 --        (*    the probabilities.                             *)
000006 --        (*****************************************************)
000006 --
000006 --
000006 --
000006 --           var
000006 --             C,P     : real;
000010 --             I,J,K,
000010 --             ROW     : integer;
000014 --
000014 --
000014 0- B       begin
004172 --             C:=0; ROW:=0;
004202 --
004202 1-          case INPUT MODE of
004205 2-           INTERACTIVE : begin
004206 --             writeln(OUTPUT);writeln(OUTPUT);
004213 --             writeln(OUTPUT,' ':12,'CUMMULATIVE PROBABILITY MATRIX');
004225 --             writeln(OUTPUT);
004230 --             writeln(OUTPUT,'ENTER PROB OF TX, CUMMULATIVE PROB WILL CALCULATED.');
004237 --             writeln(OUTPUT);
004242 --
004242 --             for I:=1 to NUM_OF_ADAPTERS do
004250 --               for J:=1 to 4 do
004256 --                 with ADAPTER[I].DEVICE[J] do
004276 --                   if not OPEN then
004302 --                     write(OUTPUT,I:5,J:1);
004322 --             writeln(OUTPUT);
004325 --
004325 --
004325 --             for I:=1 to NUM_OF_ADAPTERS do
004333 --               for J:= 1 to 4 do
004341 --                 with ADAPTER[I].DEVICE[J] do
004361 --                   if not OPEN then
004365 3-                    begin
004365 --                      writeln(TERMINAL,I:2,J:1);
004376 --                      ROW:=ROW+1;
004402 --                      C:=0;K:=1;
004405 --                      while K<=D_NUM do
004411 4-                       begin
004411 --                        read(TERMINAL,P);
004415 --                        if (C+P) <= 1 then
004424 5-                         begin
```

```
791   004424 --                    PR[ROW,K]:= C+P;
792   004444 --                    C:=C+P;
793   004450 --                    K:=K+1;
794   004454 -5                   end
795   004454 --                  else
796   004455 --                   writeln(OUTPUT,'** ERROR ** REENTER LINE, TOTAL PERCENT CANT BE > 1.0 ');
797   004465 -4                 end; (* while K <= D_NUM *)
798   004470 --                 readln(TERMINAL);
799   004473 -3          end;  (* not OPEN *)
800   004504 -2         end; (* I NTERACTIVE case *)
801   004505 --
802   004505 2-        FILE_INPUT   : begin
803   004506 --          for I:= 1 to NUM_OF_ADAPTERS do
804   004514 --            for J:= 1 to 4 do
805   004522 --              with ADAPTER[I].DEVICE[J] do
806   004542 --                if not OPEN then
807   004546 3-               begin
808   004546 --                 ROW:=ROW+1;
809   004552 --                 for K:= 1 to D_NUM do
810   004563 --                   read(DESCRIP_FILE,PR[ROW,K]);
811   004607 --                 readln(DESCRIP_FILE);
812   004612 -3               end; (* for I:= 1 D_NUM *)
813   004623 --
814   004623 -2         end; (* FILE case *)
815   004624 --
816   004624 -1        OTHERWISE : end;
817   004635 --
818   004635 --
819   004635 -0 B       end;  (* procedure to initialize cummulative prob matrix *)
820   004637 --
821   004637 --
822   004637 0- A  begin  (* procedure CHARACTERIZE NETWORK *)
823   004637 --
824   004637 --      writeln(TERMINAL,'Input mode:  INTERACTIVE or FILE');
825   004653 --        readln(TERMINAL,RESP);
826   004661 --        if (RESP[1] = 'I') or (RESP[1] = 'i') then
827   004671 --          INPUT_MODE := INTERACTIVE
828   004671 --         else
829   004675 --          INPUT_MODE := FILE_INPUT;
830   004677 --
831   004677 --
832   004677 --    (*       Begin description of network by describing adapters     *)
833   004677 --
834   004677 1-      CASE INPUT_MODE OF
835   004702 --
836   004702 2           INTERACTIVE  : begin
837   004703 --
838   004703 --            writeln('*******************************************************');
839   004712 --            writeln('          *****************************************');
840   004721 --            writeln;
841   004724 --            writeln('                  DESCRIPTION OF NETWORK');
842   004733 --            writeln('            DEVICE BY DEVICE -- ADAPTER BY ADAPTER');
843   004742 --              writeln;
844   004745 --
845   004745 --            for I:= 1 to NUM_OF_ADAPTERS do
846   004753 3-             begin
847   004753 --               with ADAPTER[I] DO
848   004764 4-                begin
849   004764 --                  writeln;
850   004767 --                  writeln('Adapter #',I:2,':');
851   005003 --                  J:=1;
852   005005 --                  while J<I do
853   005011 5-                   begin
854   005011 --                     writeln(TERMINAL,' ':2,'Propagation distance in sec to adapter ',J:2,' is ');
855   005032 --                     readln(TERMINAL,PROP_DIST_TO[J]);
856   005045 --                     J:=J+1;
857   005051 -5                  end;  (* while J<K *)
858   005054 --
859   005054 --                  if J=I then J:=J+1;
860   005063 --
861   005063 --                  while (J>I) and (J<=MAX_NUM_ADAPTERS) DO
862   005072 5-                   begin
863   005072 --                     writeln(TERMINAL,' ':2,'Propagation distance in sec to adapter ',J:2,' is ');
864   005113 --                     readln(TERMINAL,PROP_DIST_TO[J]);
865   005126 --                     J:=J+1;
866   005132 -5                  end;  (* while J>I AND <= MAX NUM OF ADAPTERS *)
867   005135 --
868   005135 --                  writeln;
869   005140 --                  writeln(TERMINAL,' ':2,'Adapter #',I:2,' fixed delay in sec is ');
870   005161 --                  readln(TERMINAL,PRIORITY_DELAY);
```

```
OC5167 --                      writeln(TERMINAL,' ':13,' end delay in sec is ');
005201 --                      readln(TERMINAL,END_DELAY);
005207 --
005227 --              (*   Begin individual device descriptions    *)
005207 --
005207 --                      for J:= 1 to 4 do
005215 5-                        begin
005215 --                          with ADAPTER[I].DEVICE[J] do
005235 6-                            begin
005235 --                                OPEN:= true;
005240 --                               writeln;
005243 --                               writeln(' ':2,'Adapter ',I:2,' Device ',J:2,' description');
005273 --                               writeln(TERMINAL,' ':4,'Device status (OPEN or CLOSED):');
005305 --                               readln(TERMINAL,RESP);
005313 --                               if (RESP[1]='C') or (RESP[1]='c') then
005325 --                                 OPEN:=false;
005326 --                               if not(OPEN) then
005332 7-                                 begin
005332 --                                   D_NUM:=D_NUM+1;
005336 --                                   DEV_NUM:=D_NUM*100 + I*10 +J;
005347 --                                   writeln(TERMINAL,' ':4,'Buffer size (Bytes) ');
005361 --
005361 --                                   (* C Ahange bytes to bits for storage *)
005361 --                                   readln(TERMINAL,BUFFER_SIZE);
005367 --                                     BUFFER_SIZE:=BUFFER_SIZE*8;
005375 --
005375 --                                   writeln(TERMINAL,' ':4,'I/O bus transfer rate',' (Bps)');
005413 --                                   readln(TERMINAL,TFER_RATE);
005421 --                                   TFER_RATE:=TFER_RATE*8;
005427 --
005427 --                                   LOAD_TIME:=1.0/TFER_RATE;
005435 --
005435 --                                   writeln(TERMINAL,' ':4,'Number of offnet sources for device:');
005447 --                                   readln(TERMINAL,NUM_OF_SOURCES);
005455 --
005455 --                                   for K:=1 to NUM_OF_SOURCES do
005467 8-                                      begin
005467 --                                        writeln(TERMINAL,' ':6,'Source #',K:2,' transmission rate',' (Bps)');
005514 --                                        readln(TERMINAL,SOURCE[K].TX_RATE);
005527 --                                        SOURCE[K].TX_RATE:=SOURCE[K].TX_RATE*8;
005547 -8                                    end;  (* for k:=1 to num of sources *)
005554 --                                   TX_INTRVL:= TX_INTERVAL(ADAPTER[I].DEVICE[J]);
005604 --                                   NEXT_TX:=TX_INTRVL;
005611 -7                              end; (* If not(OPEN) *)
005611 -6                            end;  (* with ADAPTER().DEVICE() *)
005611 -5                        end;  (* for J=1 TO 4 *)
005616 -4          end; (*  wwith ADAPTER() *)
005616 -3      end; (* for I=1 to num of adapters *)
005623 --      INITIALIZE PROB MATRIX;
005630 -2  end; (* INTERACTIVE case *)
005631 --
005631 2-  FILE_INPUT  : begin
005632 --
005632 --      reset(DESCRIP FILE);
005640 --      for I:= 1 to NUM_OF_ADAPTEPS do
005646 3-     begin
005646 --        with ADAPTER[I] DO
005657 4-          begin
005657 --            J:=1;
005661 --            while J<I do
005665 5-              begin
005665 --                readln(DESCRIP_FILE,PROP_DIST_TO[J]);
005700 --                J:=J+1;
005704 -5              end;  (* while J<K *)
005707 --
005707 --            if J=I then J:=J+1;
005716 --
005716 --            while (J>I) and (J<=MAX_NUM_ADAPTERS) DO
005725 5-              begin
005725 --                readln(DESCRIP_FILE,PROP_DIST_TO[J]);
005740 --                J:=J+1;
005744 -5              end;  (* while J>I AND <= MAX NUM OF ADAPTERS *)
005747 --
005747 --            readln(DESCRIP FILE,PRIORITY DELAY);
005755 --            readln(DESCRIP_FILE,END_DELAY);
005763 --
005763 --        (*   Begin individual device descriptions    *)
005763 --
005763 --            for J:= 1 to 4 do
005771 5-              begin
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
 951   005771 --                       with ADAPTER[I].DEVICE[J] do
 952   006011 6-                         begin
 953   006011 --                           readln(DESCRIP_FILE,RESP);
 954   006012 --                           if (RESP[8]='F') or (RESP[8]='f') then
 955   006027 --                             OPEN:=false
 956   006031 --                            ELSE
 957   006035 --                             OPEN:=TRUE;
 958   006040 --                           if not(OPEN) then
 959   006044 7-                             begin
 960   006044 --                               readln(DESCRIP_FILE,DEV_NUM);
 961   006052 --                               D_NUM:=DEV_NUM DIV 100;
 962   006060 --                               readln(DESCRIP_FILE,BUFFER_SIZE);
 963   006066 --
 964   006066 --                               readln(DESCRIP_FILE,TFER_RATE);
 965   006074 --
 966   006074 --                               LOAD_TIME:=1.0/TFER_RATE;
 967   006102 --
 968   006102 --                               readln(DESCRIP_FILE,NUM_OF_SOURCES);
 969   006110 --
 970   006110 --                               for K:=1 to NUM_OF_SOURCES do
 971   006122 8-                                 begin
 972   006122 --                                   readln(DESCRIP_FILE,SOURCE[K].TX_RATE);
 973   006135 -8                                end;  (* for k:=1 to num of sources *)
 974   006142 --                              TX_INTRVL:= TX_INTERVAL(ADAPTER[I].DEVICE[J]);
 975   006172 --                              NEXT_TX := TX_INTRVL;
 976   006177 -7                           end; (* if not(OPEN) *)
 977   006177 -6                        end;  (* with ADAPTER().DEVICE() *)
 978   006177 -5                      end;  (* for J=1 TO 4 *)
 979   006204 -4              end; (*  wwith ADAPTER() *)
 980   006204 -3         end; (* for I=1 to num of adapters *)
 981   006211 --         INITIALIZE PROB MATRIX;
 982   006216 -2        end; (* FILE_INPUT case *)
 983   006217 --
 984   006217 -1      otherwise  : end; (*  case  *)
 985   006230 --
 986   006230 -0 A   end; (* procedure CHARACTERIZE NETWORK *)
 987   006232 --
 988   006232 --    (*******************************************************)
 989   006232 --    (*                                                      *)
 990   006232 -- A       procedure PRINT NET DESCRIPTION;
 991   000006 --    (*          prints the description of the entire        *)
 992   000006 --    (*          network.                                    *)
 993   000006 --    (*******************************************************)
 994   000006 --
 995   000006 --    var
 996   000006 --      I,L,J,K,
 997   000006 --      LL,ROW    : integer;
 998   000014 --
 999   000014 --
1000   000014 0- A  begin   (* procedure PRINT DESCRIPTION *)
1001   006232 --
1002   006232 --     ROW:=0;
1003   006241 --
1004   006241 --
1005   006241 --    rewrite(AUX_OUT);
1006   006246 --    for I:= 1 to  NUM_OF_ADAPTERS   do
1007   006254 1-     begin
1008   006254 --       NEWPAGE;
1009   006260 --
1010   006260 --      writeln(AUX_OUT,' ':10,'**********************************************************');
1011   006272 --      writeln(AUX_OUT,' ':10,'           ****************************************');
1012   006304 --      writeln(AUX_OUT);
1013   006307 --      writeln(AUX_OUT,' ':10,'                         NETWORK DESCRIPTION');
1014   006321 --      writeln(AUX_OUT,' ':31,'Adapter #',I:3);
1015   006336 --      writeln(AUX_OUT);
1016   006341 --      writeln(AUX_OUT,' ':10,'           ****************************************');
1017   006353 --      writeln(AUX_OUT,' ':10,'**********************************************************');
1018   006365 --      LCT:=LCT+9;
1019   006371 --
1020   006371 --          with ADAPTER[I] DO
1021   006402 2-            begin
1022   006402 --              writeln(AUX_OUT);writeln(AUX_OUT);
1023   006407 --              writeln(AUX_OUT,'Adapter #',I:2,':');
1024   006423 --              J:=1;
1025   006425 --                LCT:=LCT+3;
1026   006431 --              while J<I do
1027   006435 3-                begin
1028   006435 --                  write(AUX_OUT,' ':4,'Propagation distance to adapter ',J:2,': ');
1029   006455 --                  writeln(AUX_OUT,PROP_DIST_TO[J]:9:7,' sec');
1030   006476 --                  LCT:=LCT+1;
```

ORIGINAL PAGE IS OF POOR QUALITY

```
1031   006502 --               J:=J+1;
1032   006506 -3             end;  (* while J<K *)
1033   006511 --
1034   006511 --           if J=I then J:=J+1;
1035   006520 --
1036   006520 --           while (J>I) and (J<=MAX_NUM_ADAPTERS) DO
1037   006527 3-             begin
1038   006527 --               write(AUX_OUT,' ':4,'Propagation distance to adapter ',J:2,': ');
1039   006547 --               writeln(AUX_OUT,PROP_DIST_TO[J]:9:7,' sec');
1040   006570 --               LCT:=LCT+1;
1041   006574 --               J:=J+1;
1042   006600 -3             end;  (* while J>I AND <= MAX NUM OF ADAPTERS *)
1043   006603 --
1044   006603 --           writeln(AUX_OUT);
1045   006606 --           LCT:=LCT+1;
1046   006612 --           write(AUX_OUT, ' ':4, 'Priority delay: ');
1047   006623 --           writeln(AUX_OUT,PRIORITY_DELAY:9:7,' sec');
1048   006637 --           LCT:=LCT+1;
1049   006643 --           write(AUX_OUT,' ':4,'End delay:       ');
1050   006654 --           writeln(AUX_OUT,END_DELAY:9:7,' sec');
1051   006670 --           writeln(AUX_OUT);
1052   006673 --           LCT:=LCT+2;
1053   006677 --
1054   006677 --   (*   Begin individual device descriptions    *)
1055   006677 --
1056   006677 --           for J:= 1 to 4 do
1057   006705 3-             begin
1058   006705 --               with ADAPTER[I].DEVICE[J] do
1059   006725 4-                 begin
1060   006725 --                   writeln(AUX_OUT);
1061   006730 --                   LCT:=LCT+1;
1062   006734 --                   write(AUX_OUT,' ':4,'Device ',J:2);
1063   006750 --                   write(AUX_OUT,' ':2,' status: ');
1064   006761 5-                   if OPEN then begin writeln(AUX_OUT,'OPEN');
1065   006773 -5                       LCT:=LCT+1; end
1066   006777 --                     else write(AUX_OUT,'CLOSED');
1067   007006 --                   if not(OPEN) then
1068   007012 5-                     begin
1069   007012 --                       writeln(AUX_OUT,' ':2,'Device number ',DEV_NUM:2);
1070   007030 --                       LCT:=LCT+1;
1071   007034 --                       write(AUX_OUT,' ':32,'Buffer size: ');
1072   007045 --                       writeln(AUX_OUT,BUFFER_SIZE DIV 8:8,' Bytes');
1073   007063 --                       LCT:=LCT+1;
1074   007067 --
1075   007067 --                       write(AUX_OUT,' ':32,'I/O bus transfer rate:');
1076   007100 --                       writeln(AUX_OUT,TFER_RATE/8:12:2,' Bps');
1077   007116 --                       LCT:=LCT+1;
1078   007122 --
1079   007122 --                       writeln(AUX_OUT,' ':32,'Load time: ',LOAD_TIME:12:7,' sec');
1080   007145 --                       LCT:=LCT+1;
1081   007151 --
1082   007151 --                       write(AUX_OUT,' ':32,'Number of offnet sources:');
1083   007162 --                       writeln(AUX_OUT,NUM_OF_SOURCES:3);
1084   007171 --                       LCT:=LCT+1;
1085   007175 --
1086   007175 --                       for K:=1 to NUM_OF_SOURCES do
1087   007207 6-                         begin
1088   007207 --                           write(AUX_OUT,' ':34,'Source #',K:2,' transmission rate: ');
1089   007227 --                           writeln(AUX_OUT,SOURCE[K].TX_RATE/8:9:2,' Bps');
1090   007252 --                           LCT:=LCT+1;
1091   007256 -6                        end;  (* for K:=1 to num of sources *)
1092   007263 --                       writeln(AUX_OUT,' ':32,'Trunk transmission interval: ',TX_INTRVL:7:4,' sec');
1093   007306 --                       LCT:=LCT+1;
1094   007312 -5                 end; (* if not(OPEN) *)
1095   007312 -4               end;  (* with ADAPTER().DEVICE() *)
1096   007312 -3             end;  (* for J=1 TO 4 *)
1097   007317 -2     end; (*  wwith ADAPTER() *)
1098   007317 -1 end; (* for I *)
1099   007324 --           NEWPAGE;
1100   007330 --           writeln(AUX_OUT,' ':6,'CUMMULATIVE  PROBABILITY OF TRANSMISSION MATRIX');
1101   007342 --           writeln(AUX_OUT);
1102   007345 --           writeln(AUX_OUT);
1103   007350 --           LCT:=LCT+3;
1104   007354 --
1105   007354 --           write(AUX_OUT,' ':5);
1106   007361 --           for I:=1 to NUM_OF_ADAPTERS do
1107   007367 --             for J:=1 to 4 do
1108   007375 --               with ADAPTER[I].DEVICE[J] do
1109   007415 --                 if not OPEN then
1110   007421 --                   write(AUX_OUT,I:6,J:1);
```

```
007441 --              writeln(AUX_OUT);
007444 --              LCT:=LCT+1;
007450 --
007450 --
007450 --              for I:=1 to NUP_OF_ADAPTERS do
007456 --                for J:= 1 to Z do
007464 --                  with ADAPTER[I].DEVICE[J] do
007506 --                    if not OPEN then
007510 1-                       begin
007510 --                         write(AUX_OUT,I:4,J:1);
007520 --                         ROW:=ROW+T;
007524 --                         for K:=1 to D_NUM do
007535 2-                           begin
007535 --                            write(AUX_OUT,PR[ROW,K]:7:2);
007557 -2                          end; (* for K := 1 to D_NUM *)
007564 --                          writeln(AUX_OUT);
007567 --                          LCT:=LCT+1;
007573 -1                     end;  (* not OPEN *)
007604 -0 A    end;    (* procedure PRINT DESCRIPTION *)
007606 --
007606 --
007606 --     (**************************************************************)
007606 --     (*                                                            *)
007606 -- A         PROCEDURE CREATE_DESCRIP_FILE;
000006 --     (*       WRITES TO THE AUXILLIARY INPUT FILE                  *)
000006 --     (**************************************************************)
000006 --
000006 --
000006 0- A     begin  (* procedure CREATE *)
007606 --
007606 --      rewrite(DESCRIP_FILE);
007620 --          for I:=-1 to NUP_OF_ADAPTERS do
007626 1-            begin
007626 --              with ADAPTER[I] DO
007637 2-                begin
007637 --                  J:=1;
007641 --                  while J<I do
007645 3-                    begin
007645 --                      writeln(DESCRIP_FILE,PROP_DIST_TO[J]);
007661 --                      J:=J+1;
007665 -3                    end;  (* while J<K *)
007670 --
007670 --                  if J=I then J:=J+1;
007677 --
007677 --                  while (J>I) and (J<=MAX_NUM_ADAPTERS) DO
007706 3-                    begin
007706 --                      writeln(DESCRIP_FILE,PROP_DIST_TO[J]);
007722 --                      J:=J+1;
007726 -3                    end;  (* while J>I AND <= MAX NUM OF ADAPTERS *)
007731 --
007731 --                  writeln(DESCRIP_FILE,PRIORITY_DELAY);
007740 --                  writeln(DESCRIP_FILE,END_DELAY);
007747 --
007747 --          (*   Begin individual device descriptions   *)
007747 --
007747 --                  for J:= 1 to 4 do
007755 3-                    begin
007755 --                      with ADAPTER[I].DEVICE[J] do
007775 4-                        begin
007775 --                          writeln(DESCRIP_FILE,OPEN);
010004 --                          if not(OPEN) then
010010 5-                            begin
010010 --                              writeln(DESCRIP_FILE,DEV_NUM);
010017 --                              writeln(DESCRIP_FILE,BUFFER_SIZE);
010026 --
010026 --                              writeln(DESCRIP_FILE,TFER_RATE);
010035 --
010035 --
010035 --                              writeln(DESCRIP_FILE,NUM_OF_SOURCES);
010044 --
010044 --                              for K:=1 to NUM_OF_SOURCES do
010056 6-                                begin
010056 --                                  writeln(DESCRIP_FILE,SOURCE[K].TX_RATE);
010072 -6                               end;  (* for k:=1 to num of sources *)
010077 -5                          end; (* if not(OPEN) *)
010077 -4                        end;  (* with ADAPTER().DEVICE() *)
010077 -3                      end;  (* for J=1 TO 4 *)
010104 -2            end; (*  wwith ADAPTER() *)
010104 -1        end; (* for I=1 to num of adapters *)
010111 --
```

```
010111 --              (* Write the probability matrix  *)
010111 --
010111 --              for I:=1 to D_NUM do
010122 1-               begin
010122 --                for J:= 1 to D_NUM do
010133 --                  write(DESCRIP_FILE,PR[I,J]);
010160 --                writeln(DESCRIP_FILE);
010163 -1              end;
010170 -0 A    end;  (* procedure CREATE *)
010172 --
010172 --
010172 --              (**************************************************************)
010172 --              (**************************************************************)
010172 --              (*                M A I N     P R O G R A M                  *)
010172 --              (**************************************************************)
010172 --
010172 0-       begin  (* main program *)
010172 --
010172 --
010172 --          (* New run initializations *)
010172 --
010172 --            for COND1 := NORMAL to OTHER do
010240 --              CONDITION[COND1] := COND1;
010260 --
010260 --             PRINT_ALL := true;
010262 --
010262 --             U:= 31622 ;  (* U is seed for RNG *)
010265 --             WAIT_TALLY:=0;
010267 --             COLLISION_TALLY:=0;
010271 --             TOTAL_ATTEMPTS:=0;
010273 --             TOTAL_ATTEMPTS:=0;
010275 --             LCT:=0;
010277 --             PCT:=0;
010301 --             TIME_ACTIVE:=0.0;
010303 --             BITS_TX:=0;
010305 --
010305 --             writeln;
010310 --             writeln(TERMINAL,'Maximum run time in secs?');
010317 --             readln(TERMINAL,MAX_TIME);
010324 --             writeln;
010327 --
010327 --             writeln(TERMINAL,'Maximum successful transmissions?');
010336 --             readln(TERMINAL,MAX_TX);
010343 --             writeln;
010346 --
010346 --             writeln;
010351 --             writeln(TERMINAL,'Seed for random number generator?');
010360 --             readln(TERMINAL,U);IU:=U;
010367 --             writeln;
010372 --
010372 --
010372 --             writeln(TERMINAL,'PRINT_ALL condition on?');
010401 --             readln(TERMINAL,RESP);
010407 --             if (RESP[1] = 'Y') or (RESP[1] ='y') then
010417 --                PRINT_ALL:= true
010417 --               else
010423 --                PRINT_ALL:= false;
010425 --
010425 --              writeln;
010430 --
010430 --
010430 --          (* Network cold start conditions *)
010430 --
010430 --             CHARACTERIZE_NETWORK;
010434 --
010434 --              page(OUTPUT);
010437 --             CREATE_DESCRIP_FILE;
010443 --
010443 --             PRINT_NET_DESCRIPTION ;
010447 --
010447 --             FIND_NEXT(TMITTR);
010454 --
010454 --             CURRENT_TIME:= TMITTR.NEXT_TX;
010457 --
010457 --
010457 --
010457 --
010457 --          (* Network steady state operation *)
010457 --
010457 1-           repeat   (* until time exceeds max time *)
```

```
010457 --              PICK_A(PCVR,TMITTR);
010465 --
010465 --              (* Calculate a collision probability *)
010465 --
010465 --                if not A_COLLISION then
010473 2-                  begin
010473 --                    TX TALLY := TX TALLY+1;
010477 --                    UPDATE_CLOCKS(CONDITION[NORMAL]);
010504 --                    if PRINT ALL then
010506 --                      PRINT_TX_STATS(TMITTR,RCVR);
010514 --
010514 --                    FIND_NEXT(TMITTR);
010521 --
010521 --                    CURRENT_TIME:=TMITTR.NEXT_TX;
010524 --
010524 -2                  end
010524 --                else
010526 2-                  begin
010526 --                    COLLISION TALLY := COLLISION TALLY +1;
010532 --                    UPDATE_CLOCKS(CONDITION[COLLISION]);
010537 --
010537 --                    if PRINT ALL then
010541 --                      PRINT_COLLISION_STATS(TMITTR,RCVR);
010547 --
010547 --                    FIND_NEXT(TMITTR);
010554 --
010554 --                    CURRENT_TIME:= TMITTR.NEXT_TX;
010557 --
010557 -2                  end;  (* if *)
010557 --
010557 -1          until (CURRENT_TIME >= MAX_TIME) or (TX_TALLY >= MAX_TX);
010571 --
010571 --          TOTAL_ATTEMPTS:=TX_TALLY+COLLISION_TALLY;
010575 --
010575 --          PRINT_NETWORK_STATS;
010601 --
010601 --          ACTIVITY_SUMMARY;
010605 --
010605 --          writeln(' END OF RUN');
010614 --
010614 -0        end.  (* main program *)
Compilation complete - no errors found.

@MAP,I  .HOSC.XQT
MAP 30R1 S74I11 07/01/83 16:39:18
START=020075, PROG SIZE(I/D)=7747/4772
SYS$*RLIB$. LEVEL
END MAP.  ERRORS: 0  TIME: 9.310  STORAGE: 12160/3/025777/073777

@XQT  HOSC.XQT

Maximum run time in secs?

Maximum successful transmissions?

Seed for random number generator?

PRINT_ALL condition on?

Input mode:  INTERACTIVE or FILE
```

```
     *** END OF RUN ***
   Current time:        36.4361 secs

   Successful transmissions:   500
   Collisions              :    29
   Waits                   :    83
   Total attempts          :   529

   Total time active       :     1.7105 secs
   % Total time active     :   4.69%

   Total Bytes transmitted :        987360

   Seed for RNG            :  312577
 END OF RUN

@BRK,Y
```

```
*** END OF RUN NETWORK STATISTICS ***

Current time:          36.4361 secs

Successful transmissions:  500
Collisions              :   29
Waits                   :   83
Total attempts          :  529

Total time active       :  1.7135225 secs
% Total time active     :  4.69%

Total Bytes transmitted :      987360

Seed for RNG            :  332577


                        DEVICE ACTIVITY SUMMARIES
                               (SECONDS)

ADP DEV    TIME           TIME        TIME IN        TIME         TIME       TRANSMISSION  RECEPTION  WAIT  COLLISION
 #   #  TRANSMITTING    WAITING     COLLISIONS    RECEIVING     ACTIVE          COUNT        COUNT    COUNT   COUNT
 1   1     0.9198        0.3897       0.0273            0       1.3357          252            0     39      20
 2   1     0.2711        0.4398       0.0045            0       0.7153           35            0     37       3
 2   2     0.4324        0.0037       0.0011       0.0012       0.4383           35            2      1       1
 3   1     0.0410        0.0110       0.0043       0.6938       0.7503           71          249      3       3
 3   2     0.0410        0.0073            0       0.3069       0.3552           71          143      2       0
 3   3     0.0052        0.0037       0.0035            0       0.0123           36            0      1       2
 4   1          0             0            0            0            0            0            0      0       0
 4   2          0             0            0       0.4376       0.4376            0           71      0       0
 4   3          0             0            0       0.2711       0.2711            0           35      0       0


                        DEVICE ACTIVITY SUMMARIES
                               (PERCENT)

ADP DEV    % TIME        % TIME          % TIME         % TIME      % TIME       % TIME IN     % TIME IN
           ACTIVE     TRANSMITTING    TRANSMITTING     WAITING     WAITING      COLLISIONS    COLLISIONS
           (TRUNK)       (TRUNK)        (DEVICE)       (TRUNK)     (DEVICE)       (TRUNK)      (DEVICE)
 1   1     78.15         5 .77            2.61          22.78        7.15           1.50          0.98
 2   1     41.82         1 .85            0.77          25.71        1.19           0.26          0.15
 2   2     25.62         25.28            1.24           0.21        0.37           0.06          0.11
 3   1     43.85          2.40            0.11           0.64        0.71           0.25          0.28
 3   2     20.77          2.40            0.11           0.43        0.71              0             0
 3   3      0.72          0.30            0.01           0.21        0.37           0.20          0.17
 4   1         0          0.30               0           0.21           0           0.20             0
 4   2     25.58             0               0              0           0              0             0
 4   3     15.85             0               0              0           0              0             0
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
          ****************************************************************
                   ********************************************

                          NETWORK DESCRIPTION
                             Adapter #  1

                   ********************************************
          ****************************************************************

Adapter # 1:
    Propagation distance to adapter  2:         0 sec
    Propagation distance to adapter  3:         0 sec
    Propagation distance to adapter  4:         0 sec

    Priority delay:           0 sec
    End delay:          0.0000010 sec

    Device  1   status: CLOSED  Device number 111
                                Buffer size:      2096 Bytes
                                I/O bus transfer rate:   500000.00 Bps
                                Load time:    0.0000002 sec
                                Number of offnet sources:  1
                                  Source # 1 transmission rate:   15000.00 Bps
                                Trunk transmission interval:   0.1337 sec

    Device  2   status: OPEN
    Device  3   status: OPEN
    Device  4   status: OPEN
```

```
          *******************************************************
                    ****************************************
                           NETWORK DESCRIPTION
                               Adapter #  2
                    ****************************************
          *******************************************************

Adapter # 2:
    Propagation distance to adapter  1:          0 sec
    Propagation distance to adapter  3:          0 sec
    Propagation distance to adapter  4:          0 sec

    Priority delay:         0 sec
    End delay:        0.0000020 sec

    Device  1   status: CLOSED  Device number 221
                                Buffer size:      4000 Bytes
                                I/O bus transfer rate:   500000.00 Bps
                                Load time:    0.0000002 sec
                                Number of offnet sources:  1
                                  Source # 1 transmission rate:   4000.00 Bps
                                Trunk transmission interval:   1.0000 sec

    Device  2   status: CLOSED  Device number 322
                                Buffer size:      6400 Bytes
                                I/O bus transfer rate:   500000.00 Bps
                                Load time:    0.0000002 sec
                                Number cf offnet sources:  1
                                  Sourc  # 1 transmission rate:   6400.00 Bps
                                Trunk transmission interval:   1.0000 sec

    Device  3   status: OPEN

    Device  4   status: OPEN
```

```
*********************************************************
        *************************************
                NETWORK DESCRIPTION
                   Adapter #  3
        *************************************
*********************************************************

Adapter # 3:
    Propagation distance to adapter  1:         0 sec
    Propagation distance to adapter  2:         0 sec
    Propagation distance to adapter  4:         0 sec

    Priority delay:          0 sec
    End delay:         0.0000030 sec

    Device  1   status: CLOSED  Device number 431
                                Buffer size:       512 Bytes
                                I/O bus transfer rate:   500000.00 Bps
                                Load time:    0.0000002 sec
                                Number of offnet sources:  1
                                  Source # 1 transmission rate:    1000.00 Bps
                                Trunk transmission interval:   0.5120 sec

    Device  2   status: CLOSED  Device number 532
                                Buffer size:       512 Bytes
                                I/O bus transfer rate:   500000.00 Bps
                                Load time:    0.0000002 sec
                                Number of offnet sources:  1
                                  Source # 1 transmission rate:    1000.00 Bps
                                Trunk transmission interval:   0.5120 sec

    Device  3   status: CLOSED  Device number 633
                                Buffer size:       624 Bytes
                                I/O bus transfer rate:  3300000.00 Bps
                                Load time:    0.0000000 sec
                                Number of offnet sources:  1
                                  Source # 1 transmission rate:     625.00 Bps
                                Trunk transmission interval:   0.9984 sec

    Device  4   status: OPEN
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
          ***********************************************************
                   *****************************************

                              NETWORK DESCRIPTION
                                 Adapter #  4

                   *****************************************
          ***********************************************************

Adapter # 4:
    Propagation distance to adapter  1:          0 sec
    Propagation distance to adapter  2:          0 sec
    Propagation distance to adapter  3:          0 sec

    Priority delay:           0 sec
    End delay:          0.0000040 sec

    Device  1    status: CLOSED   Device number 741
                                  Buffer size:      2000 Bytes
                                  I/O bus transfer rate:  3300000.00 Bps
                                  Load time:    0.0000000 sec
                                  Number of offnet sources:  0
                                  Trunk transmission interval:   10000000.3499 sec
    Device  2    status: CLOSED   Device number 842
                                  Buffer size:      2000 Bytes
                                  I/O bus transfer rate:  1200000.00 Bps
                                  Load time:    0.0000001 sec
                                  Number of offnet sources:  0
                                  Trunk transmission interval:   10000000.3499 sec
    Device  3    status: CLOSED   Device number 943
                                  Buffer size:      2000 Bytes
                                  I/O bus transfer rate:  1200000.00 Bps
                                  Load time:    0.0000001 sec
                                  Number of offnet sources:  0
                                  Trunk transmission interval:   10000000.3499 sec
    Device  4    status: OPEN
```

CUMMULATIVE PROBABILITY OF TRANSMISSION MATRIX

```
          11      21      22      31      32      33      41      42      43
  11       0       0       0    0.70    1.00    1.00    1.00    1.00    1.00
  21       0       0       0       0       0       0       0       0    1.00
  22       0       0       0       0       0       0       0    1.00    1.00
  31       0    0.01    0.02    0.02    1.00    1.00    1.00    1.00    1.00
  32       0    0.01    0.02    1.00    1.00    1.00    1.00    1.00    1.00
  33       0       0       0       0       0       0       0    1.00    1.00
  41       0       0       0       0       0       0       0       0       0
  42       0       0       0       0       0       0       0       0       0
  43       0       0       0       0       0       0       0       0       0
```